Universidade de Aveiro
2007
Departamento de Electrónica, Telecomunicações e Informática

Hugo Luís de Melo Pais

Portal de Informação Biomédica para Doenças Raras

Universidade de Aveiro
2007
Departamento de Electrónica, Telecomunicações e Informática

Hugo Luís de Melo Pais

Portal de Informação Biomédica para Doenças Raras

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia de Electrónica e de Telecomunicações, realizada sob a orientação científica do Professor Doutor José Luís Oliveira, Professor Associado do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro.

o júri

presidente

Doutor Joaquim Arnaldo Carvalho Martins
Professor Catedrático da Universidade de Aveiro

Doutora Maria do Céu Gomes dos Santos

Doutor José Luís Oliveira
Professor Associado da Universidade de Aveiro

agradecimentos

Agradeço a todos os colegas do grupo de Bioinformática, em especial aqueles com que colaborei mais directamente, pelo apoio e amizade, fazendo com que este desafio corresse de uma forma suave e sem percalços. Foi um prazer conviver com este grupo de pessoas fantásticas.

palavras-chave

Integração de dados, Bases de dados biomédicas, Doenças genéticas raras, Genética, Genómica, Web 2.0, Webcrawlers

resumo

Os avanços mais recentes nas áreas da genómica e proteómica têm resultado num crescimento significativo em termos de informação disponibilizada em fontes públicas. Espera-se que toda esta informação possa ser integrada na prática clínica, onde diagnósticos e tratamentos passarão a ser suportados ao nível molecular.

Contudo, navegar através de bases de dados de genética e bioinformática, pode ser uma tarefa muito complexa e improdutiva para um clínico. Tanto mais, considerando o campo das doenças geneticas raras, onde foi verificado que o conhecimento está concentrado num pequeno grupo de peritos.

Neste trabalho partiu-se de uma plataforma já existente, DiseaseCard.org, na qual foram introduzidas novas funcionalidades ao nível de extracção de conhecimento sobre doenças raras. Criaram-se interfaces intuitivos usando características Web 2.0, modelos de navegação capazes de guiar o utilizador no manancial de informação genética e de fenótipos e ainda um sistema de fóruns de comunicação onde os utilizadores podem partilhar experiências.

keywords Database integration systems, Biomedical databases, Genetic rare diseases, Genetics, Genomics, Web 2.0., WebCrawlers

abstract The most recent advances in the areas of genomics and proteomics are resulting on a significant growth in the available information in public databases. It is expected that all this information can be integrated in the clinical practice where diagnosis and treatments will be supported at a molecular level.
However, navigating through the databases can be a complex and unproductive task for a clinician especially on the field of rare diseases, where it was verified that the knowledge is concentrated in a small group of experts.
In this work we have started from an existing platform, DiseaseCard.org, in which there developed new functionalities, namely graphical interfaces using Web 2.0 characteristics, navigation models capable of guiding the user through a huge source of genetic and phenotype information and also communication forums where users can share their experiences.

Índice

Índice de Figuras

Introdução

1.1 Enquadramento

A sequênciação de genomas de vários organismos, incluindo o humano, levou ao aparecimento de novas áreas científicas – a genómica funcional, genómica comparativa e evolutiva, proteómica e bioinformática – que têm como objectivo estudar o funcionamento e a evolução dos seres vivos ao nível molecular e de um modo global. Estas novas áreas estão também na base de uma das maiores revoluções de sempre na medicina e na farmacologia. Existem actualmente inúmeras bases de dados públicas na área da genómica e da medicina. No entanto, a sua utilização não está ao alcance de qualquer utilizador quer porque se tratam de sistemas com diferentes mecanismos apresentação e procura de informação, quer porque o seu próprio conteúdo está tipicamente desfasado de um enquadramento mais alargado que cobre o cenário "da doença ao gene".

1.2 Objectivos

O objectivo deste trabalho era o de continuar o desenvolvimento de um portal para a integração de informação clínica e genética sobre doenças raras (DiseaseCard .org). Para tal pretendia-se construir um motor de recuperação de informação que fosse dinamicamente configurado e no qual seja descrito, de um modo formal, o protocolo de navegação sobre um conjunto predefinido de bases de dados públicas.

O utilizador acede ao sistema indicando uma determinada doença (ou por pesquisa). Identificada a doença o portal recolhe e constrói uma estrutura de *hiperlinks* organizada de acordo com o protocolo definido (sintomas, órgãos, patologia, centros, mutações, fármacos, SNP, estrutura proteica, testes genéticos, etc.) permitindo ao utilizador aceder directamente à informação sobre cada um destes subtemas.

Genericamente, os objectivos principais deste trabalho consistiram em:

- Melhorar o modelo de navegação sobre bases de dados públicas, que incluem informação sobre: sintomas, órgãos, patologia, centros, mutações, fármacos, SNP, estrutura proteica, testes genéticos, etc.
- Avaliar e melhorar a usabilidade do portal desenvolvendo novas facilidades.
- Construir um sistema de gestão do portal para aferir sobre a sua utilização, índices de carga, registo de falha, entre outros, de forma a melhor perspectivar a sua evolução futura.

1.3 Estrutura da dissertação

Para além deste, a dissertação divide-se em mais quatro capítulos.

No capítulo 2 serão apresentados todos os conceitos *Web* 2.0 e sua principais características. Serão descritas as principais tecnologias catalogadas como *Web* 2.0 e a tecnologia principal de desenvolvimento, o *Ajax*.

No capítulo 3 é apresentada uma perspectiva actual sobre o que é um sistema de *Webcrawling*, caracterizando os seus fundamentos teóricos abordando a sua perspectiva histórica e introduzindo e discutindo as técnicas utilizadas para a construção destes sistemas. Será também apresentado um pequeno estudo comparativo sobre as diversas soluções de código livre actualmente disponíveis para a comunidade científica.

No capítulo 4 é apresentada a descrição do funcionamento do sistema *DiseaseCard* e também o trabalho desenvolvido, desde os melhoramentos de interface e usabilidade, aos componentes de administração do sistema.

No capítulo 6 são apresentadas as conclusões, são discutidas direcções prospectivas do *DiseaseCard* , bem como uma auto-avaliação sobre os conhecimentos adquiridos durante a realização do projecto.

2 *Web* 2.0, o que é e o que representa?

Caracterizar a *Web* 2.0 é uma tarefa algo difícil. Estamos perante uma revolução na forma como usamos a Internet? Será uma nova bolha tecnológica? Actualmente a discussão em volta dos conceitos da *Web* 2.0 está muito activa existindo respostas diferentes conforme os quadrantes académicos e as experiências de cada utilizador. A referência mais facilmente aceite por um grande número de utilizadores é a associação do conceito a um grande número de tecnologias, como por exemplo *wikis*, *podcasts*, etc, tecnologias que vieram facilitar ligações sociais dentro da *Web* permitindo a edição e partilha de informação.

Um dos autores que se opõe ao conceito da Web 2.0 é precisamente aquele que é considerado o pai da Internet Tim Berners-Lee, argumentando que tem um sentimento de *déjà vu*. À pergunta: *"A Web 2.0 é diferente da que pode ser chamada Web 1.0 porque a anterior é tudo sobre ligar as pessoas?"* colocada num *podcast* da IBM terá respondido[1]: *"Totalmente não. A Web 1.0 é toda sobre ligar as pessoas. Era um espaço interactivo, e eu penso que a Web 2.0 é claro uma peça de jargão, que ninguém sabe o que realmente significa. Se a Web 2.0 são blogs e wikis, então isso é de pessoa para pessoa. Mas isso é o que a Web deveria ter sido desde sempre. E de facto, você sabe, esta Web 2.0, utiliza todos os standards que foram produzidos por todas as pessoas que trabalharam na Web 1.0"*.

Para perceber a resposta dada por Berners-Lee há que compreender o historial no desenvolvimento da *Web*. Durante o período em que trabalhava no CERN, Berners-Lee desenvolveu uma ferramenta colaborativa, um gestor de projectos muito rudimentar, denominado como *Enquire* que permitia que notas fossem ligadas entre elas e editadas. No entanto, com os avanços verificados à data tanto em hardware como em software, o desenvolvimento levou a criação da *World Wide Web* e à construção de uma aplicação cliente que permitia ver e editar páginas em linguagem *HTML* – "Mosaic"[2] (que mais

[1] Laningham(ed.), developerWorks Interviews,22nd August, 2006
[2] http://pt.wikipedia.org/wiki/Mosaic

tarde veio a dar origem ao navegador conhecido por "Netscape"). O facto de apenas ser possível editar conteúdos usando determinados portos levou as pessoas a considerarem a www, como um meio em que apenas alguns conseguiriam publicar e em que a maioria poderia visualizar. Na realidade o comportamento foge ao que era pretendido por Berners-Lee que tinha uma visão, explanada no livro *Weaving the Web (1999)*, em que a *Web* era um espaço de trabalho colaborativo em que tudo estava ligado entre si e em que todos poderiam editar esse mesmo espaço.

A terminologia *Web* 2.0 foi inicialmente introduzida por Dale Dougherty[3], vice-presidente da *O'Reilly Media Inc*. durante uma discussão de equipa que visava a criação de uma potencial futura conferência sobre a *Web*[4]. A equipa pretendia, com essa conferência, perceber os caminhos de desenvolvimento da *web* após a explosão e subsequente queda das empresas *dot-com*, principalmente depois de se ter verificado que regularmente apareciam novas aplicações e páginas e que as empresas que haviam sobrevivido ao decaimento de *dot-com*'s estavam mais fortes e possuíam inovações e comportamentos muito similares entre elas. Durante essa conferência Tim O'Reilly [1] e um grupo de discussão composto também por Paul Graham e Dion HinchCliffe tentaram identificar o que diferenciava a *Web* 2.0 e a *Web* 1.0. Este processo revelou-se complexo, não sendo fácil classificar alguns dos produtos identificados como *Web* 2.0 como, por exemplo, o *Napster* ou *BitTorrent*, que não são propriamente aplicações *Web*. Após o debate chegaram à conclusão que, como a maioria dos conceitos importantes, a *Web* 2.0 não poderia ser encarada como uma bolha fechada pois assim dificilmente se conseguiria desenvolver o conceito. Como resposta a esta problemática foi escolhida a abordagem baseada num núcleo gravitacional em que um conjunto de regras e princípios se encontram ligados como ilustração desse universo e foi desenhado um mapa de mnemónicas capaz de reproduzir os princípios chave da *Web* 2.0 (Figura 1).

[3] http://www.oreillynet.com/pub/au/26
[4] http://www.web2con.com/web2con/

Figura 1 – Mapa de mnemónicas da *Web* 2.0 desenvolvida na conferência da *O'Reilly Media*. Mostra as ideias que radiam do núcleo da *Web* 2.0.

Desta discussão resultam sete princípios base para definir o que é a *Web* 2.0:

1. A *Web* como uma plataforma;
2. Aproveitar a inteligência colectiva;
3. Os dados são o novo "*Intel Inside*";
4. Fim do ciclo de liberação de software;
5. Modelos de programação leves;
6. Software acima de um único dispositivo;
7. Experiências de utilização ricas.

Estes princípios base são capazes de envolver uma quantidade enorme de produtos e são tipicamente complexos pois possuem diversas interpretações possíveis. Nos próximos pontos serão descritos os três que considero mais importantes para perceber o fenómeno *Web* 2.0, procurando explicá-los segundo a visão obtida após o estudo de diversos autores e a minha percepção destes.

2.1 Aproveitar a inteligência colectiva

Actualmente a produção de conteúdos está francamente facilitada. Um utilizador pode adicionar um vídeo ou uma foto no seu espaço digital, etiquetá-lo com palavras-chave adequadas e fazer com que estes dados estejam disponíveis para os seus amigos e para todo o mundo. Simultaneamente, milhões de outros utilizadores estarão a criar e a editar *blogs* e a trabalhar para criar informação através do uso de *wikis*. Este comportamento veio alterar rapidamente os meios de comunicação, tendo estes iniciado reestruturações muito grandes na forma como obtêm notícias e conteúdos, devido ao facto de os consumidores se estarem a tornar contribuintes para programas, jornais e sites. Este fenómeno deve-se fundamentalmente ao facto de os utilizadores terem acesso generalizado a recursos digitais como câmaras digitais de alta qualidade, telemóveis e telefones inteligentes (*smartphones*), Internet, computadores e software. O fenómeno do "Jornalista Cidadão" está a escalar de uma forma tão grande que os principais jornais criam pontos de contacto directo com os leitores. Por exemplo o jornal britânico "*The Sun*[5]" tem um telefone público sempre disponível para receber todas as "histórias" que o público possa gerar. Outro exemplo, é o serviço noticioso OhmyNews [6] que diz possuir um exército de 40000 jornalistas cidadãos e 50 jornalistas profissionais que editam as notícias geradas pelos primeiros. Um dos projectos mais interessantes ligados a esta área foi realizado pela BBC [7] no seu "*Creative Archive*" que permite que um utilizador veja e faça uso de conteúdos antigos pertencentes ao canal, podendo usá-los para produzir variantes ou mesmo novos conteúdos para televisão.

Embora esta atitude participativa e criativa, gerada pelo efeito colectivo seja considerada por uma grande maioria um benefício, existem diversos autores cépticos e cautelosos relativamente a este rumo. Por exemplo, corremos o risco de perder o sentido de autoridade e estrutura de um jornal editado, onde o processo de selecção e reflexão é obrigatório [2]. Apesar do fenómeno ser indiscutivelmente de grande escala existem diversos autores que refreiam o nível de preocupação de uns e excitação de outros

[5] http://www.thesun.co.uk/article/0,,11028-10088,00.html
[6] http://english.ohmynews.com/
[7] http://creativearchive.bbc.co.uk/

argumentando que no serviço de *blogs Blogger*[8], 10 dos 13 milhões de blogs criados se encontram inactivos.

O termo "Aproveitar a inteligência colectiva" foi usado inicialmente por Tim O'Reilly [1] e logo depois surgiram diversas questões sobre a utilização da palavra inteligência: Que sentido terá? A que género de inteligência se refere? Podemos fazer uma relação directa "informação" → "inteligência" mas essa também não é a abordagem mais correcta. O'Reilly associou-a ao conceito de Sabedoria de Multidões (*Wisdom of Crowds*), conceito introduzido por James Surowiecki no seu livro intitulado *The Wisdom of Crowds* [3]. O autor definia a problemática em três tipos distintos de problemas que ele denominava de: Cognição, Coordenação, Cooperação.

Os conceitos introduzidos por Surowiecki são muito populares entre os autores que publicam sobre conceitos relacionados com a *Web* 2.0, sendo que uma grande maioria tenta adaptar os modelos apresentados por Surowiecki às observações na *Web* e nas actividades relacionadas com a *Web*.

Tim O'Reilly usa o exemplo da empresa Cloudmark[9] que agrega um filtro de *spam* colaborativo e que utiliza como aprendizagem a classificação dos utilizadores quanto ao que é ou não *spam*. O sistema Cloudmark obtém melhor desempenho na filtragem de correio electrónico do que sistemas que dependem da análise da estrutura do correio. Outro exemplo interessante é o caso do programa televisivo "Quem quer ser milionário?" em que o concorrente tem a possibilidade de questionar o público sobre a resposta a uma pergunta. Existe aqui um comportamento individual associado ao grupo, em que o grupo tem maior probabilidade de ter a resposta correcta. Estes dois sistemas implementam uma arquitectura de participação para recolher a inteligência de grupo.

Uma das questões mais interessantes levantadas por estas linhas de pensamento reflecte-se na percepção de quem possui o conhecimento e como este é obtido.

Um termo importante na definição da inteligência colectiva é também o *Crowdsourcing*, termo introduzido pelo jornalista Jeff Howe[10] para definir um processo de *outsourcing* na *Web* para a procura de conteúdos, pequenas tarefas e até soluções de problemas científicos. O conceito de *Crowdsourcing* é usado ao seu nível mais básico em sites de partilha de dados multimédia como o Youtube e o Filckr entre outros.

[8] http://www.blogger.com
[9] http://cloudmark.com/
[10] http://crowdsourcing.typepad.com/

Outros dos exemplos são os das companhias InnoCentive[11] e YourEncore[12], que propõem nos seus sites uma série de problemas colocados pelos seus clientes para qualquer pessoa resolver, em que o primeiro indivíduo a encontrar a solução para o mesmo problema recebe um prémio que pode chegar aos milhares de dólares.

Outro termo que tem grande impacto na definição da inteligência colectiva é *Folksonomy*, definido por Thomas Vander Wal[13], como resultado da sua experiência no desenvolvimento de sistemas taxionómicos em ambientes comerciais.

Vander Wal [4] definiu *Folksonomy* como o resultado de um indivíduo etiquetar informação e objectos (qualquer coisa com hiperligação) para seu próprio acesso, em que o resultado da etiquetação é feito num ambiente social, ou seja, aberto para outros mas o acto de etiquetar é feito pelo indivíduo que consome a informação. O acto não é colaborativo nem pode ser considerado uma forma de categorização, ou seja, os indivíduos não estão tanto a categorizar, nem a fornecer meios para ligar informação estão apenas fornecer o que significa na sua percepção [4]. Então o valor da *Folksonomy* é derivado de três elementos chaves de dados:

- Etiquetagem pessoal;
- O objecto a ser etiquetado;
- E a etiqueta a ser agregada ao objecto.

Como todos os utilizadores vão gerar etiquetas, as etiquetas vão ser potencialmente repetidas tornando possível inferir da análise destas escolhas os interesses dos utilizadores e permitindo às empresas adequar os produtos aos consumidores.

2.1.1 Os dados são o novo *Intel Inside*

Na Idade da informação em que vivemos a quantidade de dados dos quais fazemos uso é cada vez maior. O'Reilly [1] discute o papel dos dados e a sua gestão em empresas *Web* 2.0, como o Google, argumentando que para esses serviços o valor do software é proporcional à escala e dinamismo dos dados que ajuda a gerir. Actualmente o Google possui uma base de dados com centenas de *peta-bytes* (10^{50}) que cresce diariamente de alguns *tera-bytes*, sendo este o efeito de rede levado ao seu expoente máximo. Muitos

[11] http://www.innocentive.com/
[12] http://www.yourencore.com/
[13] http://iavoice.typepad.com/ia_voice/2006/08/interview_with_.html

dos dados recolhidos por empresas como o Google, Amazon ou Ebay são recolhidos indirectamente dos utilizadores, sendo os dados usados para a aprendizagem do sistema.

Mas esta capacidade de integração de dados e conhecimento, apesar facilitar a vida a todos os utilizadores, conduz a questões sobre um lado mais negro inerente à quantidade de conhecimento disponível: Quem é o dono dos dados? Quais são as implicações na privacidade dos utilizadores?

2.1.2 O Efeito de Rede

O efeito de rede é geralmente um termo usado na economia para descrever o aumento no valor, para os utilizadores, de um serviço no qual existe um tipo de interacção com outros, à medida que mais pessoas o começam a utilizar [5]. O efeito de rede é geralmente usado na área de telecomunicações para descrever a extensão do aumento, em utilidade, de um sistema de telecomunicações.

Na área da computação social existe um paralelo claro com tecnologias como o MySpace e Hi5, podendo-se fazer a analogia que à medida que um utilizador se junta a um determinado serviço, todos os outros utilizadores beneficiam do facto de ele aderir. À medida que o efeito de rede cresce, aumenta também a popularidade do serviço podendo mesmo gerar um efeito de bola de neve. Um exemplo recente deste efeito verifica-se no Youtube que teve um aumento de 297% de utilizadores únicos entre Janeiro e Julho de 2006 passando de 4,9 milhões para 19,6 milhões apenas nos Estados Unidos da América [6] .

Um efeito perverso que o efeito de rede pode ter é tornar referência produtos ou tecnologias de inferior qualidade, sendo um exemplo muito comum a disputa tecnológica entre o sistema *VHS* e o *Betamax*. O fenómeno assume uma dimensão social propagando-se de "boca em boca", sendo um factor fundamental de divulgação na sociedade [5]. Torna-se igualmente fundamental estabelecer normas que garantam a interoperabilidade de sistemas [7].

2.2 Principais serviços e aplicações da *Web* 2.0

Existem diversos serviços e aplicações que demonstram as fundações do conceito *web* 2.0. Estes serviços incluem *Blogs*, *Wikis*, serviços de partilha de dados multimédia, distribuição (*syndication)* de informação, *podcasting* e etiquetagem de conteúdos.

Nesta secção é feita uma pequena revisão aos serviços mais utilizados, de forma a relacioná-los com os conceitos apresentados no ponto 2.

2.2.1 Sistema Wiki

O sistema *Wiki* é uma página ou um conjunto de páginas *web* que podem ser facilmente editadas por qualquer utilizador ao qual seja permitido acesso [8]. Um caso de sucesso é a *Wikipédia*[14] significando que o conceito de produção colaborativa é bem compreendido pelos utilizadores. As páginas *wiki* possuem um botão de edição que permite a um utilizador aceder a uma ferramenta de edição simples e fácil de utilizar, podendo editar e até mesmo eliminar informação introduzida por outros. Para tal usa também uma linguagem de hipertexto simplificada permitindo criar navegabilidade entre conjuntos de páginas. As *wikis* usualmente possuem capacidade de *rollback*, ou seja de restaurar versões anteriores. A sua flexibilidade e acesso aberto são duas das razões do seu sucesso e da sua utilidade no que concerne a trabalho de grupo [8]. Começam a ser igualmente populares em ambiente empresarial para grupos de trabalho com acesso restrito [9]. Na Tabela 1 são apresentados alguns exemplos de *Wikis* e de software para a criação destas.

Tabela 1- Exemplos de *Wikis* e software para criação de *Wikis*

Exemplos de Wikis:
http://wiki.oss-watch.ac.uk/ - *Wiki* sobre aconselhamento de software de fonte aberta
http://www.wikihow.com/ - *Wiki* sobre "Como fazer… ?"
Software para criação de Wikis:
http://www.twiki.org/- *Wiki* empresarial & plataforma de colaboração
http://meta.wikimedia.org/wiki/MediaWiki - *Wiki* para media

[14] http://pt.wikipedia.com

2.2.2 Os Blogs

O termo *web-log* ou *Blog*, foi atribuído por *Jorn Barger*[15] em 1997, tendo iniciado a introdução de pequenos comentários e *links* no seu site *Robot Wisdom*[16] tornando-se assim um pioneiro do *blogging*. Um *Blog* consiste numa página *web* simples composta com pequenos parágrafos de opinião, informação, denominados *posts*, ordenados cronologicamente com o mais recente primeiro num formato de um diário *online* [10] sendo que a maioria dos *blogs* permitem aos visitantes adicionar comentários a uma entrada do *blog*.

Um dos fenómenos mais interessantes dos *blogs* é a capacidade que eles vieram atribuir a cada indivíduo para produzir conteúdos no imediato, editando as suas páginas em tempo de jornalismo, enquanto que anteriormente o conceito de páginas *web* era mais próximo de um conceito de relatório (trabalho escolar) em que o tempo de criação e publicação não era imediato [11].

A criação de hiperligações é um aspecto fundamental do *blogging* e deriva directamente da natureza de conversação da denominada blogoesfera. Também facilita a extracção de informação e a sua classificação em *blogs* distintos. Para isso os *blogs* utilizam:

- *Permalinks* – são hiperligações internas geradas pelo sistema de *blogging* e são aplicados a cada *post* particular. Se o *post* for movido ou renomeado o *permalink* mantém-se sempre o mesmo.
- *TrackBack* – também conhecido por *pingback*, este componente permite a um *blogger* A notificar um *blogger* B que referenciaram ou comentaram os *posts* do *blogger* B. Muitas vezes estes sistemas são deliberadamente desactivados por forma a eliminar caminhos para *spammers*.
- *Blogroll* – é uma lista de *links* para outros *blogs* que um utilizador adiciona ao seu *blog* por forma a referenciar outros que considere úteis.

Actualmente uma grande parte dos *blogs* existentes também permite *Syndication* fornecendo *RSS* e *Atom*. Outra tendência é o aparecimento de *photo-blogs* e *vídeo-blogs*. Na Tabela 2 são apresentados exemplos de *blogs* e software de criação de *blogs*.

[15] http://en.wikipedia.org/wiki/Jorn_Barger
[16] http://www.robotwisdom.com

Tabela 2- Exemplos de blogs e software para criação de blogs

Exemplos de blogs:
http://radar.oreilly.com/
http://www.techcrunch.com/
Software para criação de blogs:
http://wordpress.org
http://www.bblog.com

2.2.3 Etiquetagem e Referência social

Uma etiqueta, ou *tag* como normalmente é conhecida, é uma palavra-chave adicionada a um objecto digital para o descrever mas não como parte de um sistema formal de classificação. A primeira ferramenta de etiquetagem (*tagging*) foi introduzida no site del.icio.us[17] de Joshua Shacter tendo iniciado o fenómeno de referênciação (*bookmarking*) social.

Segundo Millen *et al* [12] os sistemas de *bookmarking* social partilham diversos componentes equivalentes, permitindo aos seus utilizadores criar listas de favoritos que podem ser guardados centralmente ou num serviço remoto, permitindo a sua partilha com outros utilizadores do sistema. As *bookmarks* também podem ser etiquetadas e podem pertencer a mais do que uma categoria, o que não acontecia em sistemas de *bookmarking* baseados em pastas.

O conceito de *tagging* foi extendido para lá da referenciação de sites, existindo agora possibilidade de referenciar uma diversidade de artefactos digitais que vai desde as fotos no serviço Filckr , vídeo no serviço Youtube e podcasts no serviço Odeo.

Actualmente verifica-se que o conceito de etiquetagem está a ser expandido para incluir as denominadas nuvens de *tags* que são grupos de *tags* pertencentes a diversos utilizadores e classificadas de acordo com o número de vezes que foram utilizadas sendo que a taxa de utilização é representada graficamente (ver Figura 2).

O potencial do *Bookmarking* social não foge ao interesse das grandes empresas estando a IBM a efectuar testes usando o sistema *DogEar tool* na sua intranet [12].

[17] http://del.icio.us/

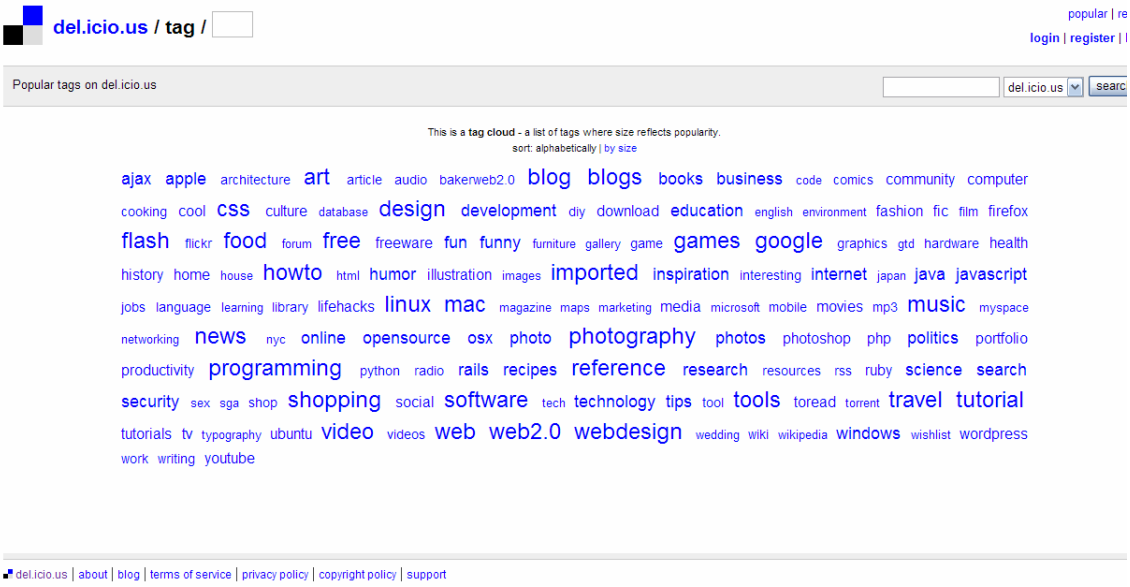

Figura 2 – Exemplo de uma nuvem de *tags* encontrada em http:// del.icio.us/tag/

2.2.4 Partilha de Multimédia

A partilha de multimédia é uma das áreas mais conhecidas, das que mais tem crescido e das que mais se tem valorizado a nível empresarial, bastando para isso observar o negócio Google com Youtube. Os exemplos mais conhecidos deste modelo são, por exemplo, Flickr, Youtube, MySpace e Odeo, que permitem que os utilizadores se tornem consumidores-produtores participando activamente na criação de conteúdos.

2.2.5 Blogging de áudio e podcasting

Os *podcasts* são cada vez mais uma marca fundamental na partilha de conteúdos entre utilizadores. São geralmente gravações áudio, tipicamente no formato *MP3*, que podem ser reproduzidas em diversos meios, desde o computador pessoal a um simples leitor portátil de *MP3*. No princípio eram denominados por áudio *Blogs* e resultaram do esforço para adicionar *streams* de áudio nos *blogs* iniciais [13]. A utilização dos áudio *blogs* explodiu com o lançamento por parte da Apple® do iPod e do software associado iTunes dando origem ao actualmente bem conhecido *podcast*. Actualmente perspectiva-se uma nova explosão com a introdução dos vídeo *podcasts*.

2.2.6 RSS e Syndication

A tecnologia *RSS* [10] (*Rich Site Summary*) é uma família de formatos que permite aos utilizadores obterem informações sobre as actualizações de *websites*, *blogs* ou *podcasts* sem ter de os visitar. Em vez disso, a informação de um site (tipicamente, um título de uma nova história e uma sinopse, juntamente com o endereço do site) é recolhida por um alimentador (*feed*) e canalizado para o utilizador num processo conhecido como *syndication*. Para um utilizador usar um *RSS* deve usar uma ferramenta denominada agregador ou leitor de *feed* no seu computador pessoal.

Em termos técnicos, um *RSS* é uma estrutura baseada num modelo de dados *XML* que permite a publicação de sumários de conteúdos. Existe um grande número de formatos *RSS*[18] (*RSS* 0.91, 0.92, 1.0 e 2.0) e com a evolução da tecnologia as últimas versões das *RSS* tornaram-se conhecidas por *Really Simple Syndication*[19].

Em 2003 surgiu um novo sistema de *syndication* que foi proposto e desenvolvido sobre o nome *Atom* para resolver problemas de consistência das RSS. O sistema *Atom* foi desenvolvido num processo formal de normalização dentro do IETF[20] (*Internet Engineering Task Force*). Em Dezembro de 2005 foi publicada a RFC4287 [14], onde é apresentada a proposta de norma do *Atom*, que é construído com dois componentes:

- Linguagem *XML* usado para os alimentadores *Web*;
- *APP* (*Atom Publishing Protocol*), que é um protocolo de comunicação baseado no protocolo HTTP para criar e actualizar os recursos *Web*.

O *Atom* faz uso de métodos HTTP simples, como o HTTP *get*, *put*, *delete* e *post* para a gestão e publicação de conteúdos de gestão. Actualmente este é usado em diversos serviços incluindo o Google, Youtube e Filckr.

[18] Para perceber a história das versões das RSS: http://blogs.law.harvard.edu/tech/rssVersionHistory
[19] Comité de aconselhamento para RSS: http://www.rssboard.org/
[20] http://www.apps.ietf.org/rfc/rfc4287.html

2.3 Principais tecnologias e Normas

2.3.1 Ajax (Asyncronous Javascript and XML)

Com o surgimento da *Web* 2.0 tornou-se necessário o desenvolvimento de técnicas de programação e de transferência de dados mais eficientes. Um dos modelos que tornou isto possível é o *Ajax* (*Asynchronous Javascript and XML*), termo criado por Garret [15], que é uma combinação de tecnologias existentes, combinadas entre si.

Com o surgimento da "tecnologia" *Ajax* foi possível resolver um dos maiores problemas que os utilizadores do modelo tradicional HTML sofriam que era os tempos de refrescamento de uma página e a quantidade de dados transferida em cada refrescamento. Como se pode observar na Figura 3 o modelo clássico de uma aplicação *Web* é composto por diversos componentes:

- o componente de suporte à interface gráfica, o *Browser* cliente,
- um meio de transporte HTTP
- e um servidor *web* com todos os componente associados a esses serviços.

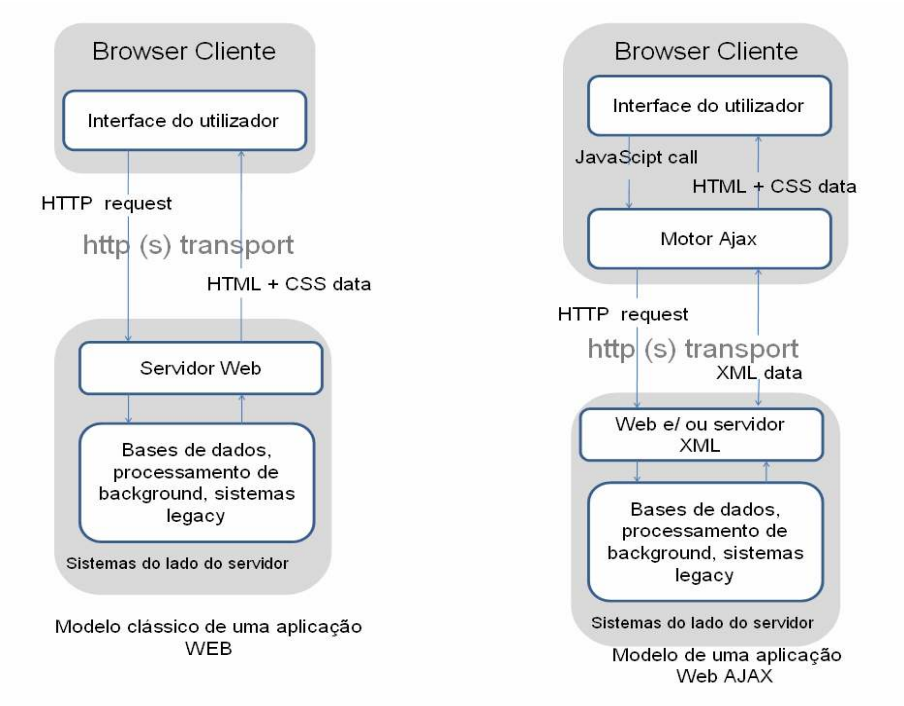

Figura 3 – Ilustração representativa dos modelos de funcionamento tradicional e *Ajax*

Neste modelo, quando o cliente deseja visualizar uma página ou altera algum formulário, o browser cliente inicia um pedido ao servidor *Web* através de um *HttpRequest*, então o servidor *Web* reconstrói novamente a página e reenvia a totalidade da mesma, sendo este processo dispendioso em tempo e tráfego quebrando a sensação de funcionamento em tempo real já tão habitual em aplicações *desktop*. No caso do modelo de uma aplicação *Ajax* existe um motor, que é um componente desenvolvido em *JavaScript*, que no caso do utilizador efectuar alguma alteração que torne necessária a actualização da página assumirá o pedido ao servidor por *postback* de uma forma assíncrona. O servidor, por seu lado, através de um serviço que gera estruturas XML, devolve uma estrutura de dados XML com a informação necessária para que a página seja parcialmente reconstruída pelo motor *Ajax* eliminando o efeito de espera e reconstrução total da página. Com a aplicação desta técnica reduz-se o tempo de resposta do servidor e a quantidade de dados transferida, bem como se melhora o desempenho do interface gráfico.

A grande diferença entre uma aplicação *Ajax* para com o modelo tradicional é o facto desta eliminar o efeito "Acção → Pausa → Acção → Pausa". Conforme representado na Figura 4, existe um intervalo temporal entre os períodos em que o cliente tem possibilidade de interagir com a página.

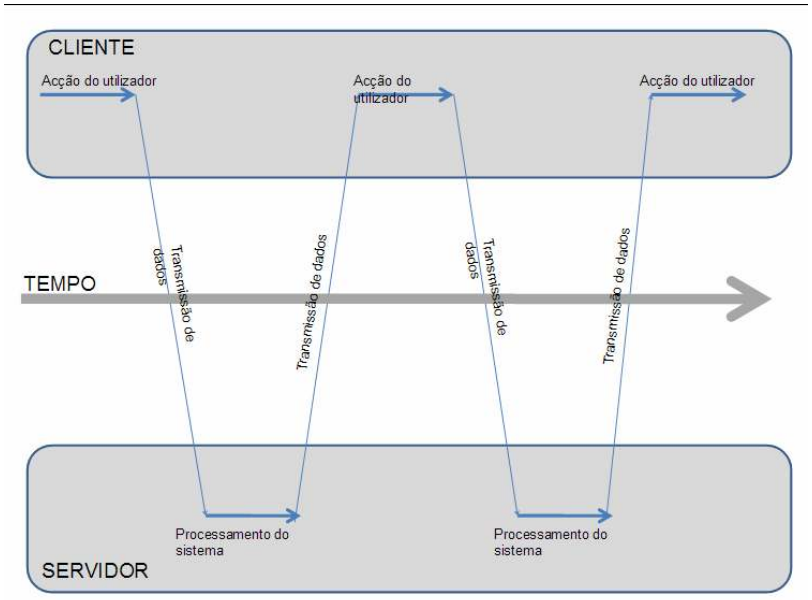

Figura 4 – Modelo clássico de uma aplicação *web* (síncrono)

Observando a Figura 5, é fácil verificar que os tempos de disponibilidade do interface é contínuo, ou praticamente contínuo, pois o motor *Ajax* comunica com o servidor através de um processo assíncrono e à medida que vai recebendo os dados vai reconstruindo o interface cliente podendo apenas actualizar zonas bem determinadas consoante os pedidos efectuados pelo cliente.

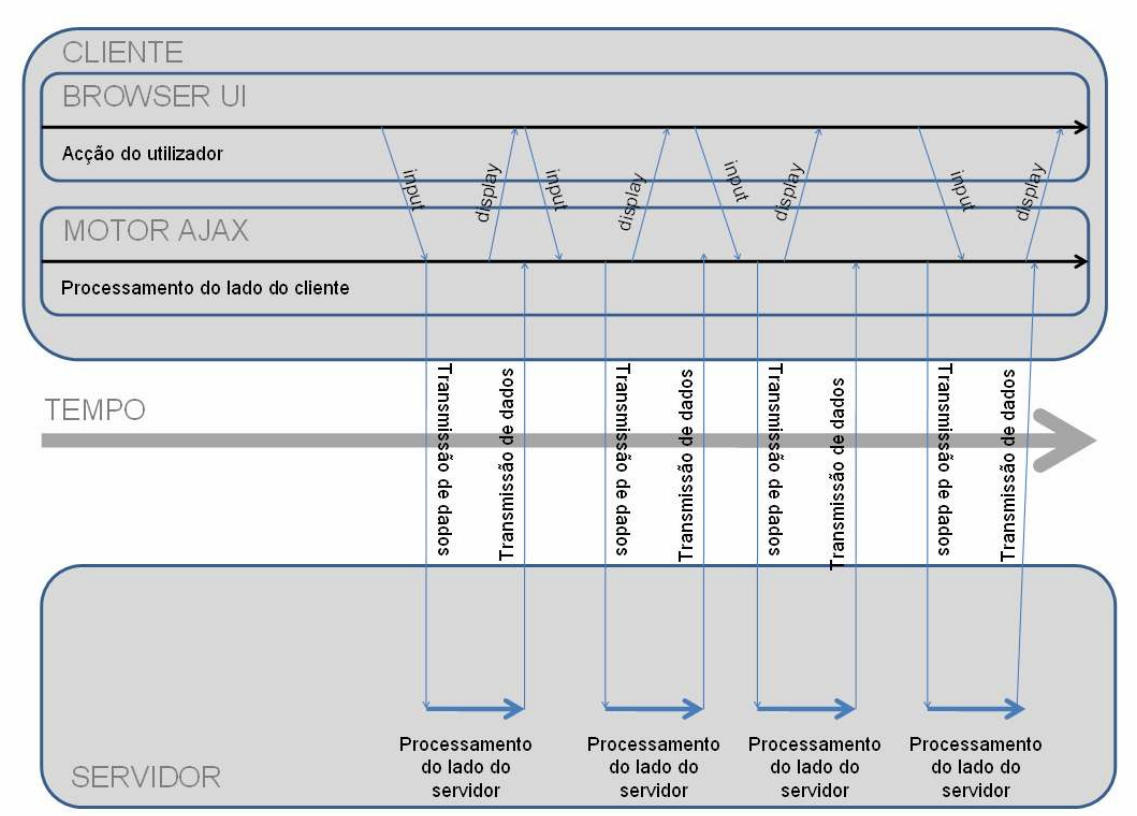

Figura 5 – Modelo Assíncrono *Ajax*

2.3.2 Alternativas ao *Ajax*

Actualmente existem diversas alternativas ao *Ajax*, sendo a mais importante o Flash, que é um *plugin* para desenho de gráficos ubíquos da *Macromedia* e tem origem nos anos 90. Actualmente é um dos motores mais utilizados para entregar *streaming* de vídeo dentro de um browser, sendo utilizado pelo *Youtube*. O Flash foi a base para outros modelos de ferramentas de desenvolvimento como o *Adobe Flex* e o *Open Lazlo*.

O ambiente de desenvolvimento *Open Lazlo* usa um modelo de fonte aberta, em que todos os programas são escritos em *XML* e *Javascript* e depois de uma forma muito transparente são compilados tanto para *Flash* como para *DHTML*.

A introdução deste tipo de modelos está a criar preocupações a diversos programadores, considerando que estas ferramentas estão a quebrar o modelo de desenvolvimento da *Web* [16] pois quebram o conceito de estrutura de hiperligações e modelo transparente de dados. Este modelo, que tornava a *Web* tão poderosa através de motores de pesquisa, corre o risco de desaparecer e as páginas *Web* transformarem-se em serviços isolados reduzindo o efeito de rede, componente fundamental da *Web* 2.0.

2.4 Sumário

Neste capítulo foram apresentados de uma forma generalista as principais características da *Web* 2.0. Actualmente a *Web* 2.0 apresenta-se como uma força motora capaz de transformar a forma como a sociedade partilha informação e conhecimento.

Com o aparecimento de conceitos como "O Efeito de Rede" e a "Inteligência Colectiva" a forma como os meios de comunicação funcionavam tiveram de ser repensados e surgiram novas oportunidades de desenvolvimento de diferentes tecnologias e metodologias. Os *blogs* e as *wiki's* apresentam-se agora como meios de partilha de conhecimento e informação por excelência, verificando-se uma descentralização do conhecimento.

Durante a discussão, foram apresentados diversos serviços classificados como *Web* 2.0, *RSS*, *Blogs*, *Wiki's*, *Podcasts*. Discutiram-se modelos de desenvolvimento de serviços *Web* inerentemente associados ao surgimento do conceito *Web* 2.0, como por exemplo o *Ajax*.

Actualmente serviços como o *Google* são chave no comportamento de utilização da *Web*, sendo que o valor deste e a sua utilidade assenta na vitalidade e qualidade dos dados recolhidos por si – "Os dados são o novo *Intel Inside*". Este último ponto serve como mote para o capítulo seguinte em que são discutidas as técnicas de *WebCrawling*, alimentadoras das bases de dados de serviços como o Google.

3 Extracção e Integração de Informação usando Webcrawling

O objectivo central deste capítulo é apresentar uma perspectiva actual sobre o que é um sistema de *Webcrawling*, caracterizando os seus fundamentos teóricos abordando a sua perspectiva histórica, e introduzindo e discutindo as técnicas utilizadas para a construção destes sistemas. Será também apresentado um pequeno estudo comparativo sobre as diversas soluções de código livre actualmente disponíveis para a comunidade científica.

3.1 Fundamentos do *WebCrawling*

Para se poder compreender de uma forma simples o que é um *Webcrawler* é necessário ter conhecimento de três pontos-chave:

- O que é a *World Wide Web* ou *web* como abreviação;
- Como é construída a informação (Qual a metalinguagem base da mesma);
- Como navegar entre pontos distintos e como se relacionam esses pontos;

Dos três pontos o que provavelmente tem uma resposta mais elementar para qualquer utilizador não especialista é o primeiro dos pontos. A *web* é uma colecção de biliões de documentos, geralmente estes têm uma dimensão de centenas a milhares de caracteres, são escritos numa diversidade de línguas e cobrem todos os tópicos de interesse dos Humanos.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
 <html lang="pt">
 <head>
  <title>Título do Documento</title>
 </head>
 <body>
  texto,
  imagem,
  links,
  ...
 </body>
 </html>
```

Figura 6 – Exemplo de texto escrito em HTML

A linguagem base de todo este manancial de informação é a *HTML* (*Hypertext Markup Language*- ver Figura 6). Esta linguagem é composta por etiquetas (*tags*), que permitem ao autor especificar uma formatação, embeber diagramas ou criar hiperligações (*Hyperlinks*) que permitem que os autores referenciem outros documentos e criem relações entre os mesmos. Os documentos só se encontram disponíveis porque existe uma infra-estrutura que permite que qualquer utilizador visualize toda a informação no seu computador pessoal, usando um *browser*. O elemento chave da infra-estrutura é o protocolo HTTP (*HyperText Transport Protocol*) construído sobre o protocolo TCP (*Transport Control Protocol*).

Após ter presentes estes três conceitos base é possível explicar o funcionamento básico de um *Webcrawler*, também conhecido como *spider*, *robot* ou *bot*.

Um *Webcrawler* explora de uma forma automática todas as páginas ou documentos presentes na *web*, recolhe-as, catalogando-as e classificando-as segundo vários itens tais como relevância e tópicos abordados. Mas como é que este consegue percorrer todas as páginas presentes na *web*? Na realidade não existe nenhum catálogo com uma listagem de todos os URL existentes na *web* e a única forma de a obter é analisar as páginas recolhidas e procurando por hiperligações que ainda não tenham sido recolhidas. Este é o princípio básico de um *Webcrawler*, ou seja, este começa por percorrer um conjunto de URL definidos à partida e, progressivamente irá recuperar e analisar novas hiperligações tornando-se esta uma tarefa potencialmente interminável. Como cada nova hiperligação significará trabalho em espera para o *Webcrawler*, esta fila de espera aumentará exponencialmente à medida que o trabalho deste prossiga, ou seja isto irá fazer com que as equipas de desenvolvimento armazenem a informação em disco por forma a libertar memória e também para evitar perda de informação em caso deste bloquear (*Crash*). Para além de hiperligações como anteriormente referido, as páginas

também possuem texto, que terá que ser classificado e indexado através de um sistema de indexação adequado, permitindo posteriormente a pesquisa de informação através de palavras-chave.

3.1.1 Síntese histórica

O conceito de *Webcrawler* começa com a publicação do artigo por Eichman em 1994 [17] . O *crawler* RBSE tinha uma arquitectura baseada em dois elementos base: um *spider* que efectuava actualização da base de dados e um *mite* que efectuava *download* dos documentos *web* e os armazenava como elementos ASCII. Mais tarde, no mesmo ano, Pinkerton [18] apresentou um sistema de *crawling* que foi usado para construir o primeiro indexador de um subgrupo da web. O *crawler* apresentado por Pinkerton foi um dos primeiros a explorar *web-graphs* e a possuir um *crawler* em tempo real. Em 1994 McBryan [19], apresentou um *crawler* que usava uma indexação simples apenas usando o título dos documentos e respectivas hiperligações. Burner [20] publicou em 1997 um artigo onde apresentava um *crawler* com algumas das características que encontramos nos *crawlers* actuais, armazenando dados parciais da web e um registo do histórico de hosts e IP's.

Em 1998 Miller *et al* [21] apresentam o primeiro *crawler* construído exclusivamente com componentes Java utilizando multitarefa e com análise gramatical do HTML. Mais tarde, nesse mesmo ano, foi publicado por alunos da Universidade de Stanford o que veio a dar origem ao colosso das tecnologias computacionais – o Google. Sergey Brin e Lawrence Page [22] apresentaram um sistema construído em C++ e *Phyton* com grande escalabilidade e com maior capacidade do que os seus antecessores. Ao momento da publicação já havia indexado 24 milhões de páginas com um repositório na ordem dos 108,7GB. O sistema tinha, para além do desempenho, uma característica inovadora: o *crawler* e o sistema de indexação e classificação são um só. Na sequência da publicação anterior é publicada por Heyden e Najork o *crawler* Mercator [23], desenvolvido em Java e muito semelhante na estrutura ao *crawler* publicado por Brin e Page. As grandes diferenças resumiam-se aos algoritmos de classificação (*page ranking*) e às metodologias de I/O, sendo que este utilizava múltiplas tarefas e *I/O* síncrono e o Google utilizava um thread único com I/O assíncrono. Esta característica tornava o Google não tão facilmente escalável. Em 2004 foi publicado outro artigo que mostra uma mudança esperada na arquitectura deste tipo de sistemas por Boldi *et al* [24]. A inovação apresentada por estes autores consistia, no facto do *crawler* ser construído

sobre uma infra-estrutura distribuída em que todas as funções se encontravam descentralizadas. Isto tornava-o escalável e ajustável às necessidades de carga do sistema, sendo comunicado pelos autores a capacidade do sistema processar 600 páginas por segundo e por computador.

3.1.2 Arquitectura Típica de um *Webcrawler*

A função central de um *Webcrawler* é recolher e processar páginas procurando hiperligações de saída e guardando-as num repositório. Tecnicamente, procurar resolver de uma forma cíclica:

- Resolução do *URL* para o endereço IP usando *DNS*;
- Ligação a um *socket* do servidor e envio do pedido de página;
- Recepção da página solicitada.

Tendo conhecimento das principais funções de um *Webcrawler* o autor Chakrabarti [25], após ter estudado a arquitectura das principais referências, Mercator e Google, propôs como anatomia base de um *Webcrawler* a Figura 7.

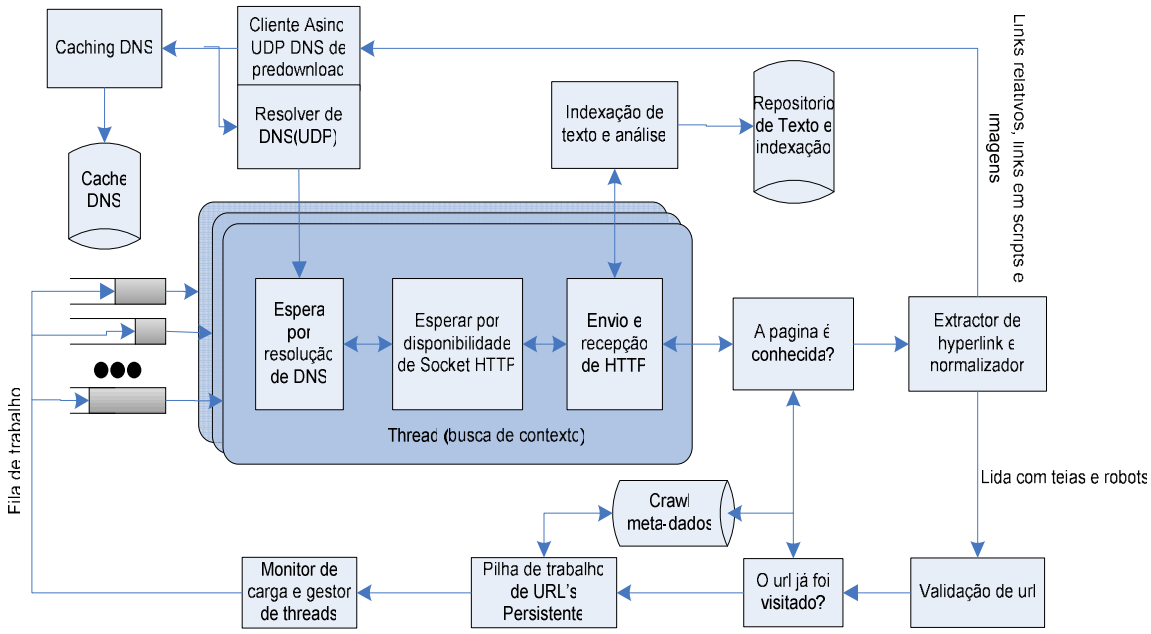

Figura 7 – Anatomia típica de um *Webcrawler* [25].

Segundo a descrição deste autor, o processo de pesquisa de domínios *DNS* e ligação aos *sockets* consomem a maior parte do tempo de processamento, dependendo dos tempos de transferência de dados.

Como tal, para uma gestão eficaz do ciclo de vida de extracção de páginas, existe um bloco de controlo lógico usando tarefas, normalmente programado especificamente para a função por forma ser obtida uma maior desempenho. No diagrama esse elemento encontra-se classificado como tarefa (busca de contexto) e o ciclo de vida deste componente começa no momento da descodificação de DNS e termina no instante em que a página é recuperada por HTTP ou então quando existe algum erro – como http 404 ou 403 (página não encontrada e acesso proibido).

Após completada a tarefa de *download*, a página normalmente é armazenada em disco e analisada procurando-se hiperligações de saída. As hiperligações de saída são reencaminhadas para uma pilha de trabalho que é gerida por um gestor de carga. Este verifica se a pilha de trabalho se encontra com dados suficientes e mantém o processamento do *crawler* a funcionar eficientemente sem criar sobrecargas, continuando o processo indefinidamente até que tenham sido recolhidos suficientes documentos, o que poderá nunca acontecer, pois cerca de 25% das hiperligações muda todas as semanas segundo Noutlas, Cho e Olston [26].

3.2 Desenho de Webcrawler's de larga escala

Quando se tratam de *Webcrawler's* de pequena escala que tem a função de analisar umas dezenas de páginas os estrangulamentos são aceitáveis e as preocupações com o design dos mesmos não são relevantes pois o tempo investido e custos de desenvolvimento superam a necessidade de desempenho. No entanto, tratando-se do design de sistema de grande escala, que têm como objectivo enviar milhões de pedidos e recolher centenas de milhões de documentos, é necessário estar muito atento a todos os pormenores que podem causar estrangulamentos ao sistema. A relevância da necessidade de um design óptimo deve-se ao facto do carregamento de uma página apenas poder significar uma latência de vários segundos, efeito agravado quando se tenta efectuar vários carregamentos simultâneos. Muitas buscas simultâneas de DNS implicam uma paralelização ou replicação de DNS. O multiprocessamento oferecido pelo sistema operativo não é o melhor para o caso em que se têm múltiplos carregamentos com grandes sobrecargas sendo igualmente necessário alguns cuidados aos extrair os URL de forma a reduzir a redundância.

3.2.1 Resolução, Pré-descodificação e *Caching* do DNS

Nos *Webcrawler's* de grande escala a resolução de endereços é o maior ponto de estrangulamento e terá que ser sobreposto a outras actividades para aumentar a desempenho do mesmo.

Um *crawler* é muito exigente na resolução de pedidos de mapeamento, podendo efectuar dezenas de pedidos por segundo. Também é bastante comuns não serem efectuados demasiados pedidos a um só servidor de forma a não sobrecarregá-lo, sendo preferível distribuir os pedidos por diversos servidores simultaneamente. Normalmente na concepção de um sistema destes é desenvolvido um DNS próprio, como exemplo o caso Mercator [23] que apresenta como resultado uma redução do tempo de resolução de domínios de 87% do tempo total de processo para 21%. Um exemplo de um bom cliente DNS aplicável a um sistema deste tipo é o ADNS que pode ser encontrado em http://www.chiark.greenend.org.uk/~ian/adns/.

Apesar destas optimizações é evidente que o problema do tempo gasto com os tempos de espera nas transferências de dados HTTP continua a ser um problema. Como tal é necessário introduzir um novo elemento que é um cliente de pré-busca (*prefectching*) usualmente implementado usando o protocolo UDP em vez de um protocolo TCP. Não esperando que a resolução seja completa, o seu comportamento serve para manter a *cache* do DNS cheia de modo a que a resolução seja mais rápida quando a página for necessária mais tarde.

3.2.2 Carregamentos Múltiplos e Concorrentes

Os *Webcrawler's* de grande escala baixam desde umas centenas de páginas até uns milhares por segundo. Um *download* de uma página pode demorar vários segundos. Para aumentar o desempenho os *crawlers* necessitam abrir o máximo de ligações a vários servidores simultaneamente. No desenho de *crawlers* de grande escala são usadas essencialmente duas técnicas distintas: multitarefa e *sockets* não bloqueantes com alimentadores de eventos.

No caso do multitarefa, após a resolução do domínio cada tarefa lógica cria um cliente de *socket*, criando uma ligação a um *socket* de um servidor HTTP e recebendo os dados enviados pelo servidor.

No caso dos *sockets* não bloqueantes com alimentadores de eventos a avaliação dos estados da ligação como ligar, enviar e receber retornam sem a necessidade da rede completar a operação o que permite que qualquer *socket* seja escrito ou lido independentemente do seu estado. É possível também associar cada *socket* a uma estrutura de dados que mantém o estado lógico da tarefa, sendo que uma centena de *downloads* concorrentes de páginas de 10Kb apenas consomem 10MB de memória.

A técnica que se apresenta como mais eficaz é a segunda pois as tarefas de *download* de páginas são seriados e o código que completa o processamento da página não é interrompido pelo término de outros processos, permitindo um acesso constante e não desordenado aos meios de armazenamento de dados. No caso de multitarefa é necessário garantir exclusão mútua e acesso concorrente a estruturas de dados, criando falhas de desempenho. Assim o término de tarefas ou processos e escrita concorrente de dados pode originar acesso aleatório de I/O no disco reduzindo o desempenho dos acessos ao disco.

3.2.3 Extracção de Hiperligação e Normalização

Em primeira análise, os URL's não parecem apresentar-se como um problema. Teoricamente, após a filtragem no processamento das páginas eram inseridos na fila de trabalho, mas, na realidade, isto não acontece e os URL's têm que ser filtrados e colocados no seu estado canónico.

A necessidade de colocar os URL's no seu estado canónico surge do facto de diversos sítios poderem ser conhecidos por vários endereços, como por exemplo: http://bioserver.ieeta.pt aponta para o mesmo sítio que http://bioinformatics.ua.pt. No caso de grandes servidores o mesmo *host* pode possuir diferentes IP e no caso de sites com muitos utilizadores este pode ter múltiplos endereços IP para balancear a carga no acesso. Existe ainda a problemática das instituições que possuem diversos sites com apenas um *host*, implementando *hosting* virtual ou passagem por *proxys*.

Para serem formados URL's canónicos [25] é necessário garantir que os seguintes passos são cumpridos:

- Uma *string* normalizada é usada para o protocolo (ex: *http*).

- O nome do *host* deve ser colocado no estado canónico, sendo isto conseguido através da utilização da resposta de um pedido de DNS que possui tanto o IP do *host* assim como o nome canónico do *host*.
- Adicionar um porto se assim for necessário.
- Normalizar todos os caminhos, eliminando todos os caminhos relativos.

Um exemplo simples do um URL apresentado no estado canónico é http://bioserver.ieeta.pt:80/DiseaseCard /index.jsp.

3.2.4 Eliminação de páginas já visitadas

Na *Web* a grande maioria dos documentos encontram-se endereçados sobre múltiplos endereços diferentes e existem casos em que os documentos se encontram distribuídos sobre *mirrors* distribuídos em múltiplos servidores. Desta forma torna-se pertinente que exista um bloco que permita evitar que seja repetido o *download* e a análise do mesmo documento, pois isso irá ter custos elevados tanto temporais como computacionais. Esse bloco é conhecido por *isUrlVisited()*. Na implementação proposta por Najork e Heydon [23] todos os URL's visitados são guardados no seu estado canónico e é mantida uma tabela de dados chamada URL Set contendo a lista mais recente de URL's visitados, estando os restantes armazenados em disco sobre a forma de um *checksum* de tamanho fixo.

A técnica implementada por Heydon e Najork [23] é a seguinte: quando é efectuado o carregamento de um documento é gerada uma impressão digital desse mesmo documento, um conjunto de 64 bits usando a implementação de Broder do algoritmo de *Fingerprinting* [27]. Em termos computacionais é guardado uma pequena *hash table* com uma lista das *fingerprints* mais recentes. Quando a *hash table* é totalmente preenchida os dados são despejados para um ficheiro e a lista mais recente reside em memória para evitar desperdícios temporais com leituras desnecessárias do sistema de ficheiros. No caso de uma impressão digital de um documento não se encontrar em memória será pesquisada no ficheiro presente em disco. Caso não seja identificada qualquer impressão digital a página é recuperada e a sua impressão digital é adicionada à *hash table*. Este processo é bastante eficaz pois a probabilidade da *fingerprint* estar na *hash table* é muito elevada.

3.2.5 Exclusão de robots

O ficheiro de exclusão de robots é um ficheiro colocado no servidor numa pasta no endereço raiz, com a função de dar informação ao *webcrawler* sobre quais as pastas a que este tem permissões de acesso e pode explorar. Este ficheiro tem a função principal de dar poder aos gestores dos conteúdos das páginas de protegerem ou evitarem que os dados publicados sejam indexados. Como exemplo e aplicação a Figura 8 apresenta regras que bloqueiam o acesso de *Webcrawler's* a algumas pastas de um determinado servidor.

```
# robots.txt
User-agent: *
Disallow:
Disallow: /DiseaseCard
Disallow:/go2gene
```

Figura 8 – Exemplo de um ficheiro robot.txt para o host http://bioinformatics.ua.pt/

3.2.6 Eliminar armadilhas (*Spider Traps*)

A defesa de um *crawler* contra armadilhas (*spider traps*) é de crucial importância pois ou por malícia ou erros de programação são geradas armadilhas impossíveis de analisar por um analisador léxico como *Flex* ou o *JFlex*. Nestas armadilhas são criados comandos capazes de gerar páginas infinitamente. Para este problema não existe remédio capaz de resolver todos os problemas, mas a análise do tamanho do URL e também a contagem dos caracteres "/" pode ser suficiente para evitar a grande maioria dos problemas de páginas com caminhos infinitos.

3.2.7 Monitorização de Carga e gestão de Tarefas

Para que o *crawler* funcione correctamente é necessária uma monitorização de carga de forma a ser possível decidir o número de *tarefas* em execução.

A monitorização de carga é efectuada em diversas estatísticas do sistema:

- Monitorização do estado de rede, usando dados relativos a latência e estimativas da largura de banda disponível.
- Monitorização de *sockets* disponíveis, sendo normal que os ISP's forneçam uma listagem com os *sockets* disponíveis para o *Webcrawler* utilizar.

Com estes dados é possível ao *crawler* fazer uma gestão eficiente dos acessos a sistemas assim como da rede à qual se encontra ligado, permitindo optimizar os recursos disponíveis e aumentar o desempenho.

3.2.8 Filas de trabalho por Servidor

Um *Webcrawler*, seja ele para trabalho de pequena ou grande escala, vai conseguir efectuar pedidos de múltiplas páginas por segundo. Um problema que acontece sucessivamente é que ao tentarmos aceder ao máximo de páginas vamos activar os sistemas de protecção do servidor nomeadamente o *denial of service* (DoS), que quase todos os servidores comerciais possuem. Esta protecção foi implementada de forma a proteger os servidores contra a saturação que é uma técnica utilizada por piratas informáticos que inundavam um servidor com pedidos de acesso a páginas normalmente usando IP's de clientes que tinham sido invadidos, provocando o bloqueio dos servidores.

A técnica utilizada por Brin e Page [22] consiste num acordo de amizade. São criadas filas de espera por *crawler*. Se o IP do servidor já foi visitado no intervalo temporal de protecção então será efectuado um tempo de espera até que se aceda ao mesmo servidor. Solução semelhante é apresentada por Heydon e Najork [23]. Com este tipo de gestão consegue-se manter o máximo de threads ocupadas sem perturbar o funcionamento dos servidores. Segundo Brin e Page [22] o Google conseguia 300 ligações em simultâneo e em picos de velocidade conseguia efectuar *download* a 100 páginas por segundo usando quatro *crawlers*. Sendo estes dados de um *crawler* de investigação certamente os dados actuais serão incrivelmente superiores.

3.2.9 Repositórios de dados

Após terem sido discutidos os pontos relativos à segurança e às metodologias de implementação necessárias para um *crawler* obter o máximo de desempenho, chega-se ao ponto em que é necessário discutir o que fazer aos dados obtidos pelo processo de extracção. O papel natural de um *crawler* é despejar todas as páginas que extrai para um repositório. Em sistemas de pequena escala é espectável que os dados a armazenar não ultrapassem o tamanho de uma unidade de armazenamento vulgar, sendo armazenada em bases de dados relacionais ou em bases de dados que efectuem armazenamento num único ficheiro. Como exemplo de uma base de dados de ficheiro único existe a base de dados *SleepyCat* actualmente conhecida como *ORACLE® Berkeley Database*. Esta base

de dados permite ser configurada como uma *hash table* ou uma *B-tree* de forma a tornar possível uma pesquisa por chave como, por exemplo, um rápido acesso por URL. Este método tem a desvantagem de que quando é necessária uma actualização da base de dados vamos ter uma perda de eficiência grande devido a uma fragmentação dos dados.

No caso dos *crawlers* de grande escala as soluções de armazenamento não podem sofrer dos mesmos problemas que as bases de dados de *crawlers* de pequena escala porque a dimensão das mesmas supera grandemente as dimensões das de pequena escala. A perda de eficiência por tempos de acesso e escrita de dados pode provocar atrasos e latência no sistema provocando um efeito em catadupa destruindo a eficiência de todo o sistema. Estes factos ilustram claramente uma regra básica de medição de eficiência: "O meu sistema é tão bom quanto o pior elemento". Para resolver tais problemas, ou pelo menos mitigar os efeitos adversos, são utilizados diversos servidores de armazenamento que usam cassetes de grande débito que conseguem números espantosos de 120MB/s. Por exemplo, o modelo T10000 da SUN® consegue numa hora armazenar 180.000.000 de páginas e ao fim de 15 horas praticamente todas as páginas da Internet, fazendo uma estimativa que todos os dados guardados ocupam em média 10Kbytes. Uma possível arquitectura é apresentada na Figura 9 em que é usado um *crawler* com acesso distribuído à *web* e suportado por um conjunto de servidores de armazenamento que guardam informação classificada por assunto, podendo até existir diversas bases de dados por cada tópico. Este é o sistema utilizado pelo Google [22] e pelo Yahoo, em que os dados estão distribuídos por várias bases de dados por tópicos de interesse.

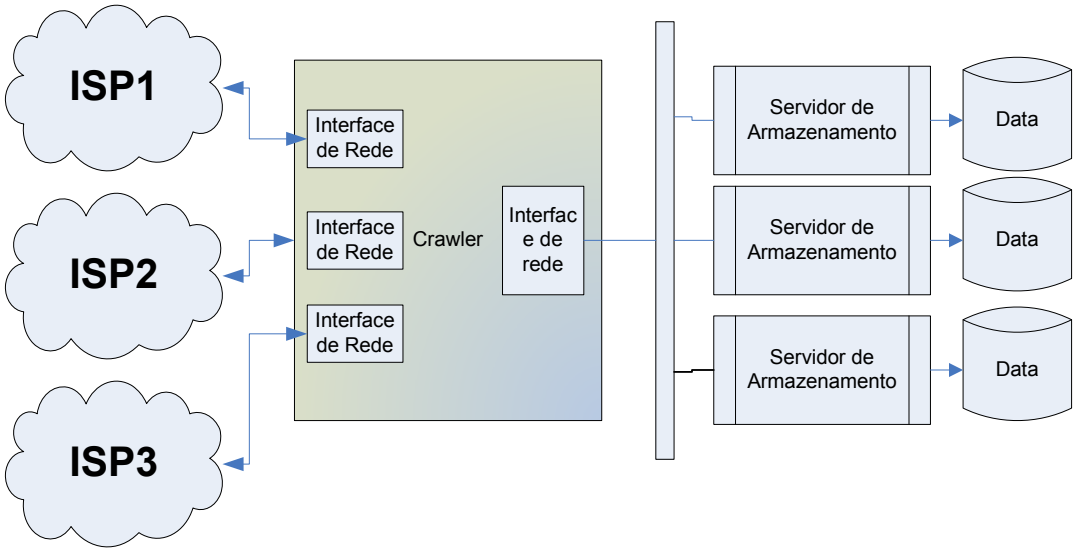

Figura 9 – Arquitectura possível de sistema de armazenamento distribuído para um *Webcrawler* de grande escala [25].

3.2.10 O que fazer com os dados recolhidos?

Para terem alguma utilidade os dados obtidos têm que ser tratados. O tratamento mais comum é aplicar técnicas de mineração de dados ou de texto com o objectivo de classificar os documentos, criando uma indexação por palavras e classificando-as por relevância. Uma técnica utilizada para classificar por relevância centra-se nos dados obtidos por monitorização de acessos a páginas, números de hiperligações a apontar para as mesmas, sendo esta a técnica utilizada pelo Google. Classificar a informação por tópicos, criar grafos de hiperligações e análise de redes sociais são outras técnicas normalmente associadas a este tipo de sistemas. Todas estas técnicas trazem grandes custos para o *crawler* pois obrigam à criação de sistemas paralelos capazes de efectuar estas análises.

No que se refere ao refrescamento dos dados, tipicamente o sistema inicia o refrescamento das páginas com mais tempo de vida na base de dados, efectuando a inserção dos URL's nas filas de trabalho do *crawler*, por forma a garantir que apenas entrem para processamento, análise e armazenamento as páginas que realmente foram alteradas substancialmente. Para isso é comparado o *fingerprint* das páginas com o que resultou do último processo de rasteio.

3.2.11 Estado de arte de *WebCrawler's* open-source

Actualmente, com o conceito de software livre, é possível encontrar diversos produtos e grande parte deles possui capacidades capazes de satisfazer as necessidades de um *crawler* de pequena escala. Como complementaridade à facilidade de utilização, o seu código é de distribuição aberta sendo possível efectuar ajustes de forma a cumprir as necessidades desejadas.

Na Tabela 3, encontram-se alguns dos *crawlers* disponíveis para utilização pública. A listagem foi ordenada com base no número de funcionalidades implementadas e o nível de informação sobre a sua utilização. Nos parágrafos seguintes são descritas algumas das características principais de alguns dos produtos listados.

Nome do Produto	Licença	Código Fonte	Linguagem	Localização
Lucene	Apache open source	Disp.	Java	http://lucene.apache.org/
webSPHINKX	Apache open source	Disp.	Java 1.2	http://www.cs.cmu.edu/~rcm/websphinx/
HERITRIX	LGPL	Disp.	Java 1.5.11	http://crawler.archive.org/
Jspider	LGPL	Disp.	Java 1.4	http://j-spider.sourceforge.net/
Webeater	LGPL	Disp.	Java	http://sourceforge.net/projects/webeater
WebLech URL Spider	open source MIT	Disp.	Java	http://weblech.sourceforge.net/
JoBo	LGPL	Disp.	Java 1.5.11	http://www.matuschek.net/jobo/
Arachnid	LGPL	Disp.	Java 1.5.11	http://arachnid.sourceforge.net/
HyperSpider	LGPL	Disp.	Java	http://hyperspider.sourceforge.net
Metis	LGPL	Disp.	Java	http://www.severus.org/sacha/metis/

Tabela 3- Listagem de *webcrawlers* de código aberto

O projecto *Lucene* (*Apache Lucene Project*) é um projecto da Apache que apresenta um protagonismo e vitalidade grandes, ganhando cada vez mais adeptos. O apache *Lucene* é uma ferramenta com um design simples é escalável, capaz de efectuar indexação com alto desempenho. Possui implementados diversos algoritmos de pesquisa, sendo outro factor interessante o facto de que possui capacidades multi-plataforma, estando desenvolvido em Java e mais recentemente em plataforma *Microsoft .Net* em C#.

Os recursos computacionais necessários para a implementação do *crawler Lucene* são extremamente baixos permitindo que este seja executado até num pequeno servidor com 1GB de memória. É capaz de obter resultados bastante interessantes, na ordem dos 20MB/minuto, ou seja em média 400 páginas por minuto, com criação de indexação e consumindo um espaço de armazenamento aproximadamente 30% do espaço original dos documentos indexados.

O *Lucene* implementa os seguintes algoritmos de pesquisa:

- Pesquisa por ranking (melhor ranking apresentado primeiro);
- Pesquisa por frase, *wildcard*, proximidade e outras;
- Pesquisa por campos (autores, conteúdos, etc);
- Pesquisa por intervalos de datas;
- Ordenação por campo;
- Pesquisa por múltiplas indexações;
- Actualização e Pesquisa simultânea.

Este projecto encontra-se muito bem documentado e tem um grande grupo de utilizadores bem com um livro de referência (*Lucene in Action*) que pode ser encontrado em *lucenebook*.com. Actualmente, considero que este produto é o *crawler open source* com maior potencial de utilização e com maior qualidade entre os projectos observados.

O *crawler* SPHINKS foi desenhado por Bahrat e Miller [28], este é uma *framework* de desenvolvimento de *WebCrawlers*, é desenvolvido em ambiente gráfico e todo o código está disponível para utilização. O *crawler* é bastante eficiente e tem implementadas as principais características de um *crawler*, sendo que as que mais se destacam são:

- Extracção de páginas *Multitarefa*;
- Modelo de objectos que representa hiperligações e páginas;
- Suporte para classificadores de páginas reutilizáveis;
- Tolerante a má formatação do *HTML*;
- Suporta o standard de exclusão de robots;
- Pesquisa de padrões usando expressões regulares;
- Concatenação de transformações de *HTML* (concatenação de páginas, salvar páginas no disco e guardar hiperligações).

O SPHINKS possui um "*workbench*" que é um GUI que permite a configuração de um *crawler* e visualização de diversos gráficos como apresentado na Figura 10 que representa uma colecção de páginas.

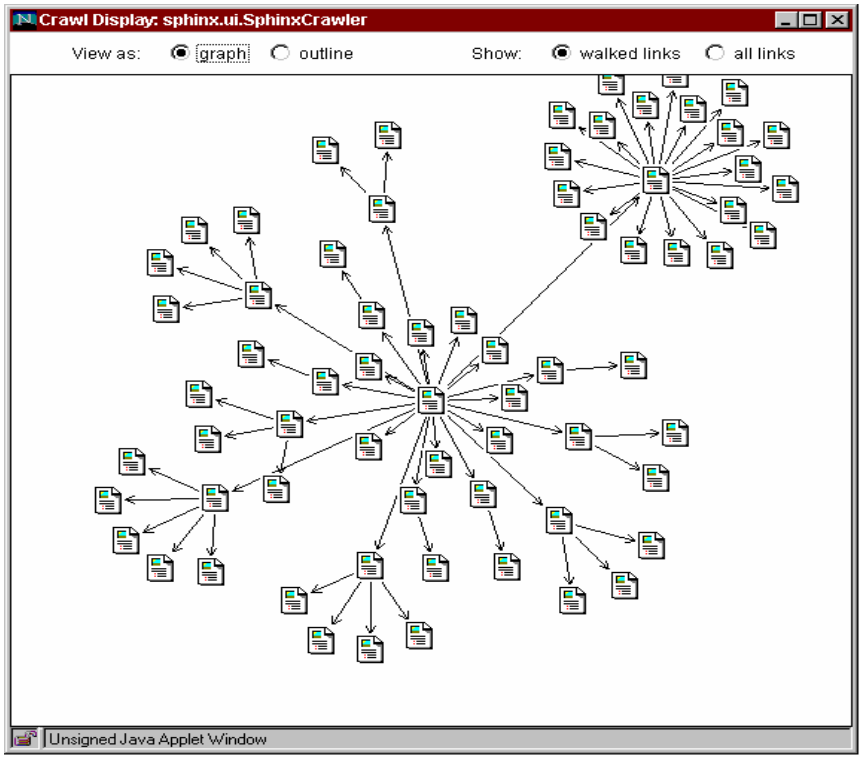

Figura 10 – Representação gráfica de mapa de hiperligações gerado pelo *crawler* SPHINKS.

O *crawler* Heritrix é uma ferramenta totalmente desenvolvida em Java que necessita do Java 1.5 para funcionar. Actualmente não existe muita informação sobre a arquitectura do sistema mas este apresenta-se como um projecto extremamente activo com actualizações muito recentes. Os autores publicaram diversos manuais tanto para utilizadores como para programadores. Como utilizadores actuais do sistema encontram-se listadas instituições como a biblioteca nacional e universitária da Islândia, biblioteca nacional da Finlândia, entre outras.

3.3 Sumário

Neste capítulo, foram apresentadas as principais características de um *webcrawler* e preocupações a ter no seu desenvolvimento. Actualmente os sistemas de *crawling* são fundamentais para a extracção de conhecimento da *Web* e para a tornar usável. Foi feita uma síntese histórica do seu desenvolvimento e descritos diversos projectos de código livre. Ao estudar a arquitectura de *Webcrawlers* ficou perceptível a quantidade de pormenores que podem ter influência no desempenho de um *webcrawler*.

4 DiseaseCard: Portal de informação biomédica

4.1 Introdução

Uma doença rara é uma doença que ocorre de forma pouco frequente na população mundial, nomeadamente com uma incidência de menos de 5 casos em cada 10000 cidadãos [29] sendo estimado que existam entre 5000 e 8000 doenças raras distintas.

Apesar da baixa incidência por doença "30 milhões de pessoas padecem de doenças raras na União Europeia" [30] . A maioria destas doenças é de difícil diagnóstico pois a maior parte dos sintomas não são identificados nas fases iniciais. Por outro lado, diagnósticos errados e tardios acabam por ser frequentes devido ao conhecimento inadequado destas doenças e aos poucos casos existentes.

Com a descodificação do código genético Humano tem existido uma grande evolução no nível de compreensão da relação entre doenças e genes, sendo que a integração de dados e tratamentos genéticos no processo clínico é cada vez mais uma realidade, permitindo diagnósticos e tratamentos facultando uma verdadeira medicina molecular. Para que tal suceda será necessária uma extensa colaboração e troca de informação entre biológicos moleculares, bioquímicos, geneticistas, clínicos, especialistas, etc.

Actualmente existem inúmeras bases de dados que permitem obter informação do domínio da relação fenótipo – genótipo. No entanto, existem enormes obstáculos à sua utilização plena pela comunidade clínica, científica ou público em geral: devido à dispersão de informação e ao facto de cada uma usar estruturas *ad-hoc*, fornecer métodos distintos de acesso à informação e criar diferentes terminologias para o mesmo conceito ou entidade. Estas especificidades levam a que apenas um pequeno grupo de investigadores altamente qualificados consiga navegar entre as bases de dados existentes, podendo mesmo entre estes existir perda de informação relevante. A importância de ter uma eficaz navegação entre bases de dados para estabelecer uma correcta relação fenótipo – genótipo é primordial no caso das doenças raras, pois cerca

de 80% destas doenças têm origem genética. Como exemplo da disparidade de informação (redundante, não certificada) pode ser dado o caso da doença de *Fabry*, uma das doenças que cumpre os requisitos de prevalência de 5 para 10000 habitantes. Quando se tenta estabelecer uma relação fenótipo – genotipo com base em pesquisas em portais populares tais como o Google obtêm−se cerca de 531000 entradas tornando inviável a tarefa de obter informação. Depois de terem sido observados tais cenários, e de forma a ser fornecida informação precisa, de grande valor científico e clínico, surgiu no seio do grupo de Bioinformática do Instituto de Engenharia Electrónica e Telemática de Aveiro (IEETA), unidade de investigação da Universidade de Aveiro, e no âmbito da rede de Excelência Europeia INFOBIOMED, o projecto *DiseaseCard* que tinha como objectivo criar um portal de informação que integrasse bases de dados distribuídas e informação heterogénea para doenças raras permitindo uma navegação transparente ao utilizador.

O projecto de investigação e desenvolvimento passou por várias fases: a) identificação das melhores fontes de informação entre bases de dados públicas; b) criação de um protocolo de navegação sobre as bases de dados; c) construção de um motor de extracção e processamento de informação d) desenvolvimento do portal *web* que associa o conceito cartão a cada doença e permite ao utilizador seguir um qualquer caminho entre a doença e o gene (Figura 11).

4.1.1 Descrição do protocolo de extracção

A Figura 12 apresenta a rede de fontes de informação que é usada actualmente no *DiseaseCard* [31] . A esta rede de fontes de informação chamamos protocolo de navegação. Para cada conceito, o utilizador tem acesso a uma ou mais fontes de dados. O ponto de entrada neste processo é uma tabela da base de dados OMIM – *MorbidMap*. Esta tabela contém a lista de todas as doenças genéticas catalogadas bem como o respectivo código OMIM e símbolo do gene caso exista.

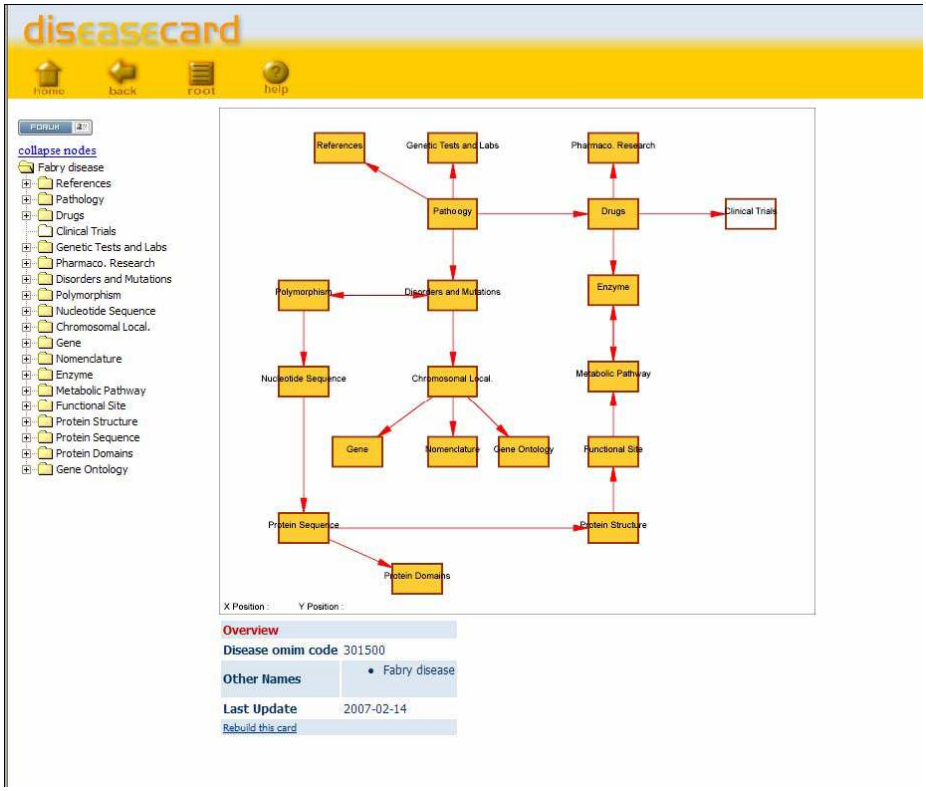

Figura 11 – DiseaseCard – vista de um cartão de doença.

O *DiseaseCard* possui esta lista armazenada localmente e é a partir desta que o utilizador inicia a consulta especificando o nome de uma doença, código OMIM ou até o símbolo de um gene para o qual pretende saber se existe alguma doença associada. Neste ponto obtém um conjunto de chaves que permitem aceder a informação sobre a patologia, fármacos existentes para a doença, mutações genéticas, etc. A partir daqui o utilizador pode percorrer toda a rede navegando para o conceito seguinte com base em informação retirada do conceito corrente. Sendo possível mapear o processo de exploração de conteúdos num protocolo de navegação e uma vez que para um dado contexto ou domínio de exploração as bases de dados a explorar são sempre as mesmas independentemente da consulta, foi desenvolvido um sistema que automatiza todo o processo de extracção e integração de informação, baseado na utilização de *web crawlers* controlados a partir de um protocolo predefinido pelos utilizadores.

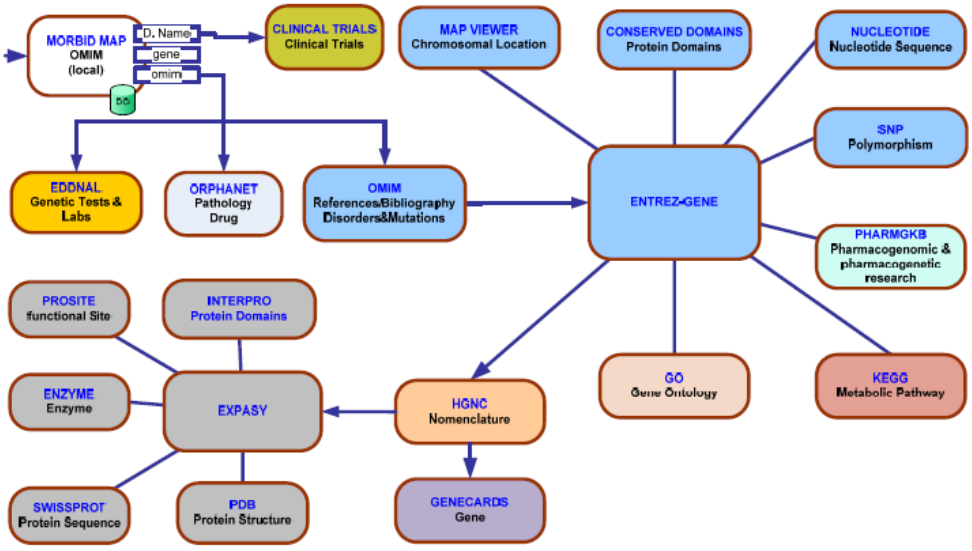

Figura 12 – [31] Exemplo de uma rede de fontes de informação associada ao contexto das doenças genéticas raras. Cada bloco representa uma fonte de dados da qual se retira informação relativa aos respectivos conceitos. Por exemplo, a base de dados *Orphanet* contém informação para os conceitos *Pathology* e *Drug*.

4.1.2 Descrição do funcionamento do DiseaseCard

O portal *DiseaseCard* [31] foi construído sobre plataforma *Web* usando *Java Server Pages* e a *framework Struts*, tendo aplicado técnicas de extracção de informação de bases de dados heterogéneas (*WebCrawlers*), apresentando vertentes não exploradas até ao momento na tecnologia, como por exemplo a capacidade de apenas lidar com *hyperlinks* (URL's) em vez de uma integração física.

A arquitectura [31] (Figura 13) está dividida em vários blocos funcionais. O *XML Protocol Descriptor* (*xpd*) representa um protocolo ou um conjunto de protocolos de navegação desenhados por investigadores com conhecimentos nos diversos domínios de extracção. Com o intuito de tornar o sistema mais flexível e abrangente estes protocolos são descritos com base numa sintaxe XML. *XML Card Descriptor* (*xcd*) é também um ficheiro XML que faz associar cada item do xpd a um nodo no mapa de representação gráfica do cartão. *XPD Engine* (XPDE) corresponde ao motor de busca de informação baseando-se nas instruções do *xpd* e nas consultas simples do utilizador comum. Este módulo é responsável por aceder a fontes externas de uma forma dinâmica e organizada. Com este mecanismo é possível alterar a descrição XML do protocolo de navegação e assim podermos criar novos protocolos os modificar os existentes sem necessidade de reescrever e compilar programas.

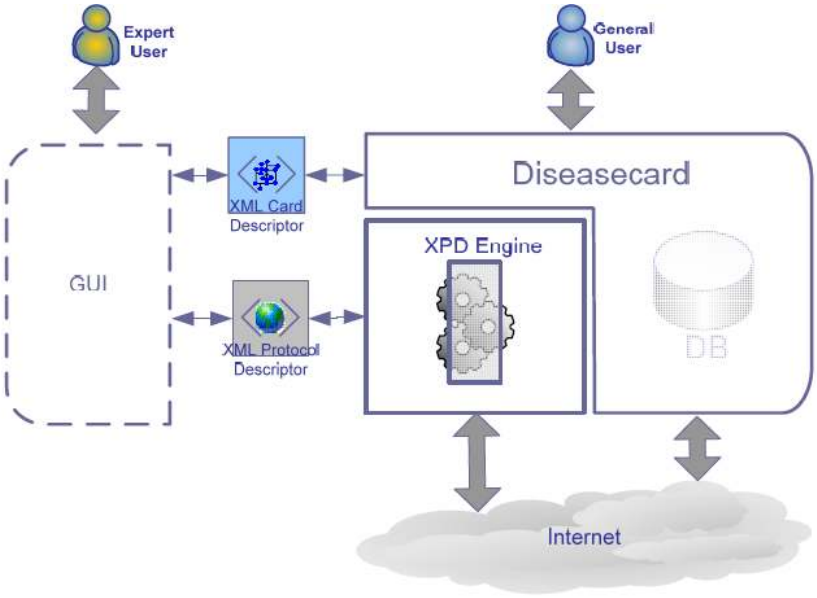

Figura 13 – Blocos principais do sistema de extracção e integração de informação proveniente de fontes de dados heterogéneas.

O bloco *DiseaseCard* corresponde à camada de aplicação que serve de interface entre o utilizador comum e o bloco funcional *XPDE*. A divisão entre estes dois blocos possibilita a reutilização do *XPDE* em diferentes aplicações. A informação é extraída com base numa consulta simples feita pelo utilizador comum que é cruzada com as instruções do protocolo especificadas no *xpd*.

Depois de o XPDE obter toda a informação, processa todos os itens de acordo com o mapeamento do ficheiro *xcd* e cria um cartão de doença. O bloco GUI (*Graphical User Interface*) é uma aplicação que servirá para criar e editar os ficheiros de configuração xpd e xcd de uma forma intuitiva para o utilizador.

4.1.3 Desafios considerados neste trabalho

Na sua versão 3.0 o utilizador delega totalmente a construção dos cartões de doença no sistema automático, evitando a necessidade de utilizadores especialistas para a criação de cartões, sendo bastante eficaz na velocidade de construção de cartões e na garantia de qualidade de informação disponível.

Como principal objectivo definiu-se o melhoramento da usabilidade do sistema, pelo que foi decidido desenvolver diversos componentes *Web* 2.0. De forma a melhorar a navegabilidade dentro do sistema foi implementado um design baseado em modelos de dados leves, *Asynchronous Javascript and XML* (*Ajax*), e integrou-se uma lista de

sinónimos capaz de referenciar praticamente todos os nomes conhecidos para cada doença, assim como abreviaturas.

Como forma de agilizar a comunicação entre comunidade em geral e a comunidade cientifica, foi criado um sistema de fórum, com entradas por doença, aberto a todos os utilizadores e moderado por um conjunto de administradores.

Foi ainda construído um sistema de administração para avaliar sobre a utilização maliciosa do sistema. Paralelamente, foi desenvolvido um sistema de actualização da base de dados abrangendo a listagem de doenças e também os seus sinónimos. É ainda possível observar o protocolo de extracção e as estatísticas de penetração do *webcrawler*.

4.2 Casos de utilização

Relativamente à versão anterior do portal, os casos de utilização foram melhorados, sendo mantidas as principais funcionalidades do sistema. A Figura 14 representa o diagrama de casos de utilização da aplicação *DiseaseCard* e as suas principais funcionalidades.

Na actual implementação do *DiseaseCard* existem quatro papéis diferentes na utilização do sistema e de relacionamento com o mesmo:

- **Utilizador Comum:** destina-se a todos os utilizadores que pretendam utilizar a aplicação *DiseaseCard* para pesquisar informação de doenças raras identificadas na base de dados do mesmo. No perfil do utilizador comum estão listados todos os utilizadores comuns assim como especialistas (ex.: geneticista, biólogo, medico de clínica geral,..)
- **Administrador *Full Access*:** A função deste administrador é garantir a actualização do sistema, gerindo a actualização da listagem de doenças e sinónimos, gerir os administradores *Full Access* e *Limited Access*, gerir a utilização dos fóruns e seu utilizadores e visualizar protocolo de extracção e estatísticas do sistema.

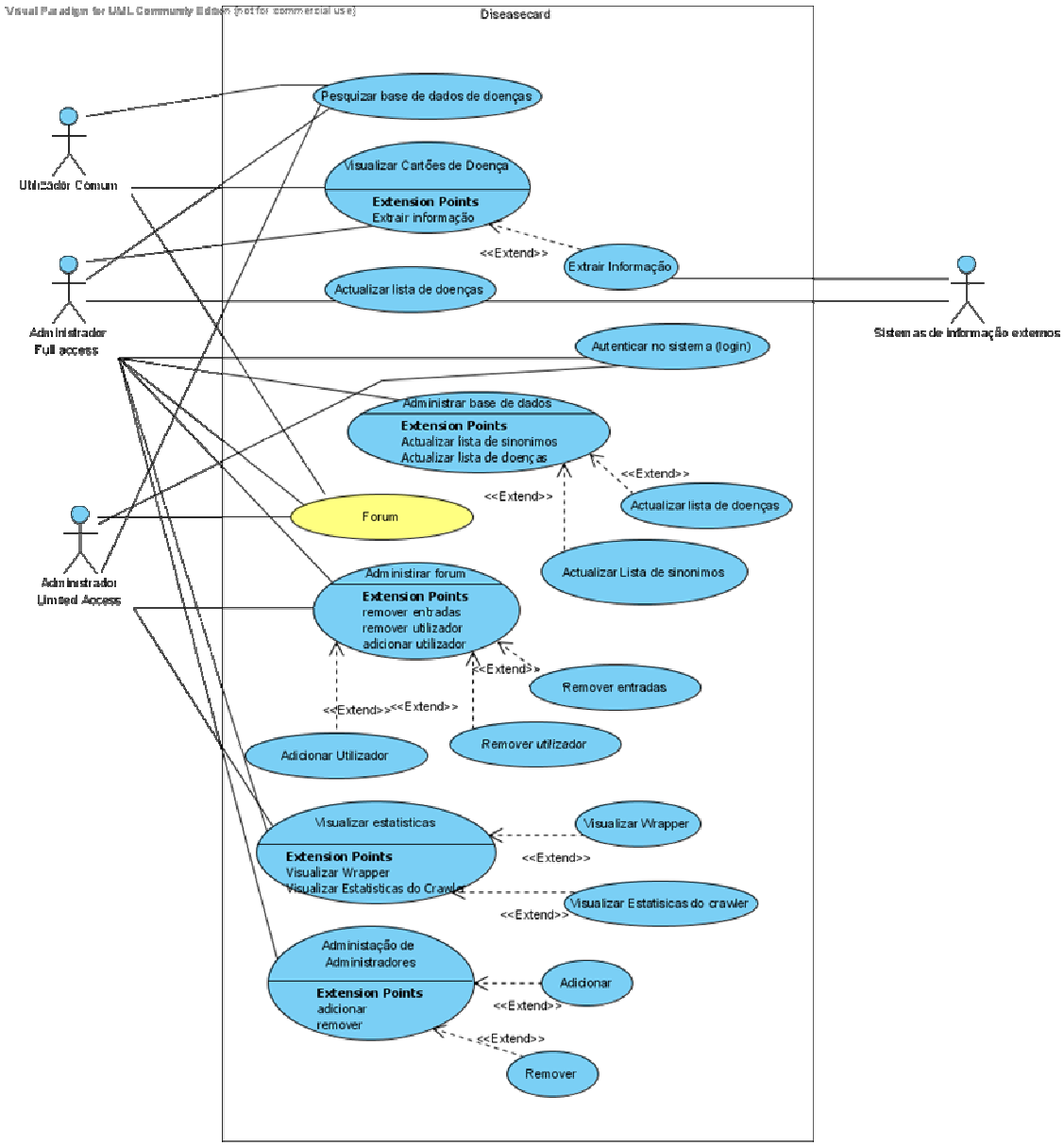

Figura 14 – Diagrama de casos de utilização da aplicação *DiseaseCard* .

- **Administrador *Limited Access*:** A funções deste administrador é colaborar na gestão da utilização dos fóruns e seus utilizadores, visualizar protocolos de extracção e estatísticas do sistema.
- **Sistemas de informação externos:** Tratam-se de todas as fontes de dados públicas sobre as quais o sistema executa consultas e extrai informação para o preenchimento dos cartões de doença.

Para estes quatro tipos de utilizadores o sistema possui as seguintes casos de utilização:

- **Pesquisar doenças:** o utilizador preenche um formulário, que é associado a um sistema de auxílio ao preenchimento, com o nome da doença da qual deseja visualizar o cartão. Além do método de pesquisa por nome de doença ou abreviaturas, é permitida a pesquisa por código OMIM ou por Gene. Para tal, o sistema consulta a base de dados retornando a lista de doenças, complementada com uma lista de nomes alternativos para cada doença.
- **Visualizar cartão de doença:** Após seleccionada a doença da lista obtida através da pesquisa da base de dados, o sistema apresenta uma lista detalhada e estruturada com os detalhes da doença organizados por conceito, apresentando uma representação em árvore ou gráfica, contendo apontadores para as fontes de dados externas que contêm informação relevante.
- **Administrar fórum:** Mediante um utilizador com um perfil de Administrador *Full Access* ou *Limited Access*, este tem a possibilidade de adicionar novos utilizadores do fórum, visualizar as entradas por cada utilizador do fórum e eliminar os utilizadores ou as suas entradas.
- **Gerir administradores:** Mediante um utilizador com perfil *Full Access* o administrador tem a possibilidade de adicionar, administradores *Full Access* e *Limited Access*.
- **Administrar base de dados ou extrair listas de doenças:** No perfil *Full Access*, o administrador tem a possibilidade de actualizar a listagem de doenças raras, através da activação de um processo de actualização da base de dados acedendo por *ftp* de uma forma automática a uma lista denominada *MorbidMap* pertencente ao site NCBI que possui a listagem de todas as doenças raras, assim como os genes envolvidos na doença, poderá também actualizar a listagem de sinónimos usando a base de dados OMIM no formato texto armazenado no servidor.

4.2.1 Sistema Ajax de suporte à pesquisa de doenças e genes

Para desenvolver o sistema *Ajax*, foi usada uma livraria de componentes *Ajax*, denominada *AjaxTags*[21]. Esta livraria é de código livre, faz uso do modelo J2EE e fornece componentes passíveis de serem usados em páginas JSP e componentes

[21] http://ajaxtags.sourceforge.net/

JavaScript capazes de garantir o funcionamento do sistema *Ajax* através de *Http requests* a ser atendidos por *Servlets*.

O funcionamento do sistema está representado na Figura 15.

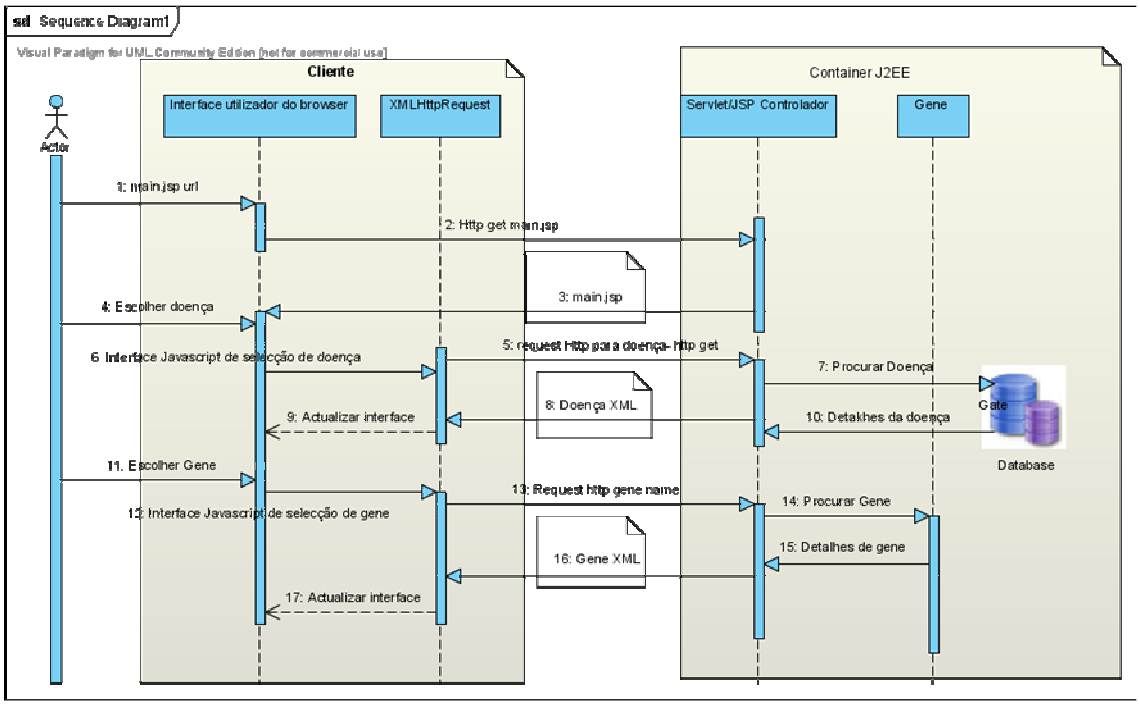

Figura 15– Diagrama de sequência do sistema de *autocomplete* usado no *DiseaseCard* .

Observando o diagrama de sequência do sistema, é simples perceber o seu funcionamento, quando o utilizador acede à página principal esta é construída no *container* J2EE, no caso do *DiseaseCard* um servidor *Apache Tomcat*, através de um pedido de get http. Logo que o utilizador fica com a página disponível no seu browser pode iniciar o preenchimento do formulário de pesquisa, introduzindo o nome da doença que pretende consultar. Com a introdução de um carácter o sistema de auxílio ao preenchimento através do interface *javascript* inicia um pedido por *Http request* de uma forma assíncrona, em *postback*, sem originar a construção total da página. Entretanto a *servlet*, que tem a função de responder aos pedidos por parte do cliente, recebe o pedido e inicia uma pesquisa na base de dados construindo uma estrutura XML com a lista de doenças que se enquadram na sequência de caracteres até ao momento inseridos pelo utilizador, após recebido os dados da estrutura XML e o interface *javascript* reconstrói a página apenas nas áreas em que existe necessidade de ser actualizada, obtendo-se o resultado exemplificado pela Figura 16.

Figura 16 – Imagem ilustrativa do sistema de auxílio ao preenchimento implementado no sistema de pesquisa do *DiseaseCard* .

4.2.2 Sistema de extracção de sinónimos e visualização

Para construir um sistema que englobasse todos os nomes possíveis e as abreviaturas conhecidas foi necessário recorrer a base de dados fornecida pelo site NCBI. Neste mesmo site encontra-se uma versão de texto da mesma base de dados. Para construir este componente foi necessário adicionar à arquitectura do *DiseaseCard* um novo componente denominado por *Parser*, conforme o representado na Figura 17.

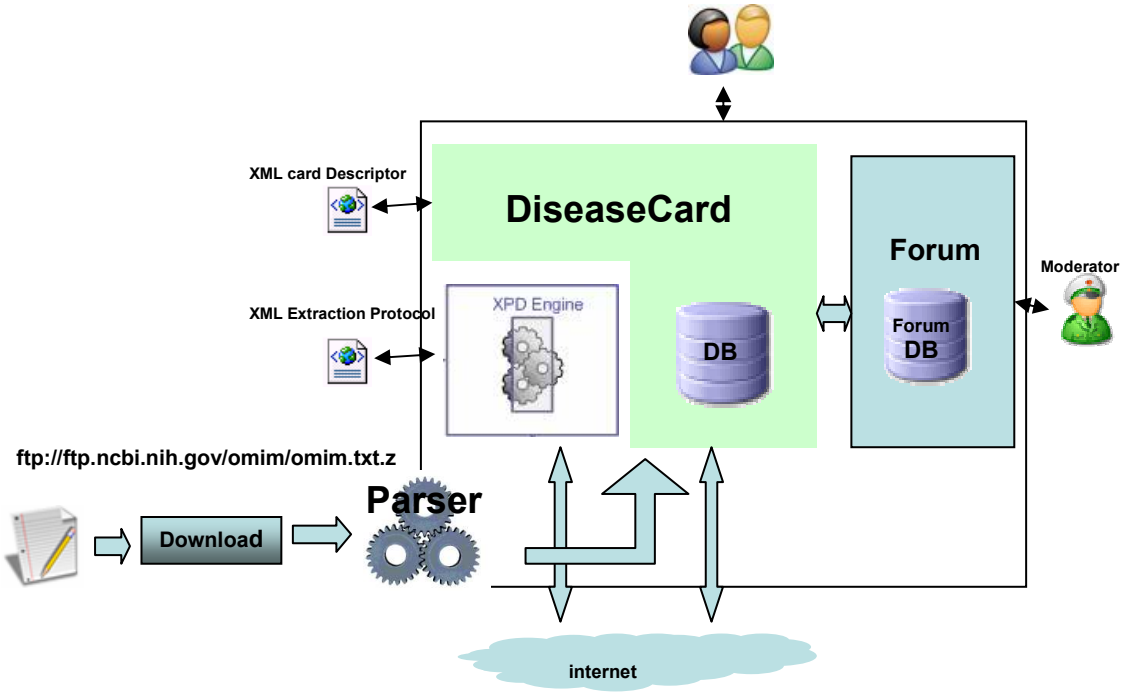

Figura 17 – Diagrama conceptual do *DiseaseCard* .

Após o *download* da base de dados, o componente *Parser* (Figura 17), através de expressões regulares e pesquisando pelos diversos campos, identifica o código OMIM que se encontra no intervalo entre o *FIELD* NO e *FIELD* TI (Figura 18). Após ter encontrado o código OMIM, é extraído o intervalo entre *FIELD* TI e *FIELD* TX deste modo é possível a separação das *strings* devido à existência de marcação por duplo ponto e virgula, assim obtendo todos os nomes e abreviaturas para a doença. Efectuada a análise estrutural, temos uma lista de nomes associados a um código OMIM, e esses têm de ser guardados na base de dados, para tal foi necessário introduzir alterações ao modelo de dados da listagem de doenças do *DiseaseCard* .

```
*RECORD*
*FIELD* NO
100070
*FIELD* TI
#100070 AORTIC ANEURYSM, ABDOMINAL
;;AAA;;AAA1;; ANEURYSM, ABDOMINAL AORTIC;;
ABDOMINAL AORTIC ANEURYSM ARTERIOMEGALY, INCLUDED;;
ANEURYSMS, PERIPHERAL, INCLUDED
*FIELD* TX
```

Figura 18 – Entrada de nomes na base de dados OMIM.txt

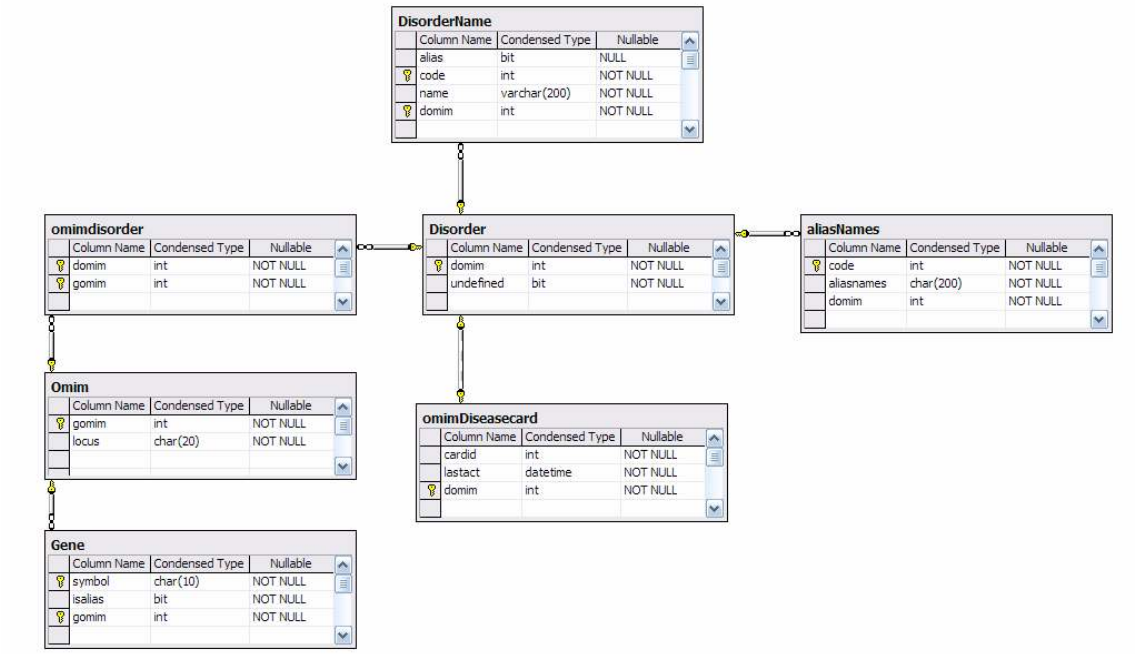

Figura 19 – Modelo de dados da base de dados de doenças do *DiseaseCard* .

Estas alterações consistiram na introdução de uma nova tabela relacional (*aliasNames*) em que a chave é o número OMIM.

Após a criação da listagem de nomes e de abreviaturas das doenças foi adicionado ao sistema a possibilidade de pesquisa por abreviaturas, resultando numa maior facilidade de pesquisa dentro do portal.

De forma a introduzir mais informação ao portal, foi criado um sistema baseado na tecnologia *Ajax*, com capacidade de listar todos os nomes da doença dentro de uma pequena janela *popup*. Esta capacidade permite ao utilizador verificar todos os nomes possíveis para a mesma doença.

O sistema funciona de uma forma semelhante à apresentada no ponto 4.2.1. Quando é gerado um evento do lado do cliente, neste caso o utilizador aponta para o identificador de informação (círculo azul com i no centro), é despoletado um pedido *Http request* que activa a *servlet* de resposta. Este, por sua vez, vai construir uma estrutura XML que será devolvida ao cliente e irá servir para preencher o conteúdo do *popup* que é gerado pelo interface *javascript* conforme o representado na Figura 20.

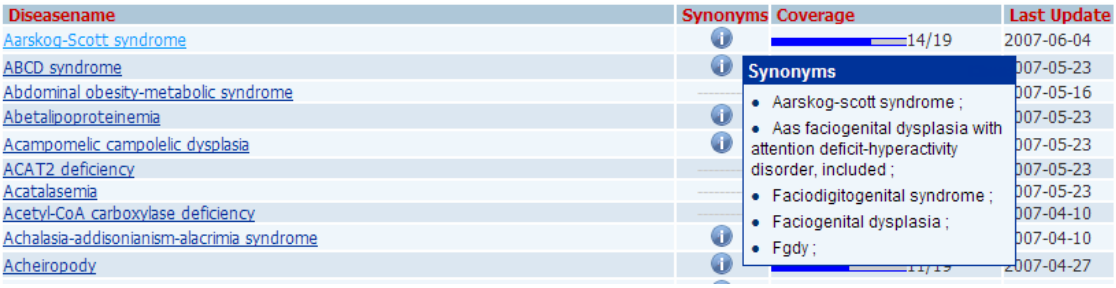

Figura 20 – Ilustração do *popup Ajax* com listagem de nomes de doença.

4.2.3 O Fórum

Para agilizar a comunicação entre a comunidade científica e a comunidade em geral, foi criado um fórum, embutido no *DiseaseCard* embora seja um que pode funcionar de um modo independente. Esta opção deve-se à necessidade de proteger o modelo de dados do *DiseaseCard* , facilitando a possibilidade de eliminar a base de dados e remover o fórum do sistema sem grandes custos de tempo de desenvolvimento.

O sistema de fórum tem apenas um tipo de utilizador (ver Figura 21), que pode abranger todos os utilizadores possíveis. O sistema é moderado mas sistema de moderação está inserido no sistema de administração. Pretendeu-se com este modelo criar um nível de autoridade igual entre utilizadores do fórum, para que não fosse possível que utilizadores eliminassem conteúdos inseridos por outros.

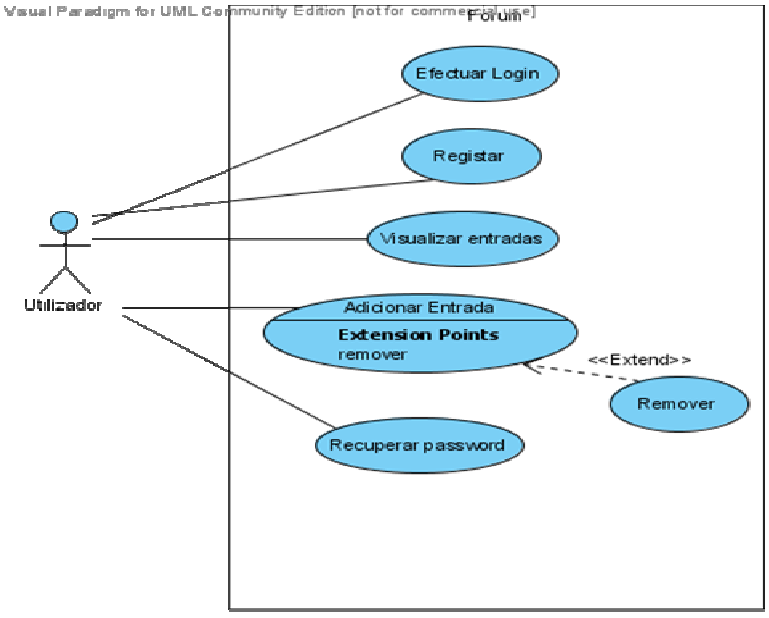

Figura 21 – Diagrama de casos de uso do fórum

O utilizador do fórum tem capacidade de registo. Quando efectua o registo, se o seu e-mail for válido, receberá um e-mail com a confirmação do registo. Caso contrário não lhe é permitido o acesso ao fórum. Após estar registado tem acesso a um sistema "*What you see is what you get*" (ver Figura 22). Para editar mensagens é usado um modelo de um editor desenvolvido em *Javascript* que permite a edição de texto HTML sem que seja necessário ter conhecimentos sobre o mesmo. É-lhe também permitido eliminar as suas próprias mensagens. No caso de perder a palavra-chave, através da inserção do e-mail usado no registo é possível recuperar a palavra-chave.

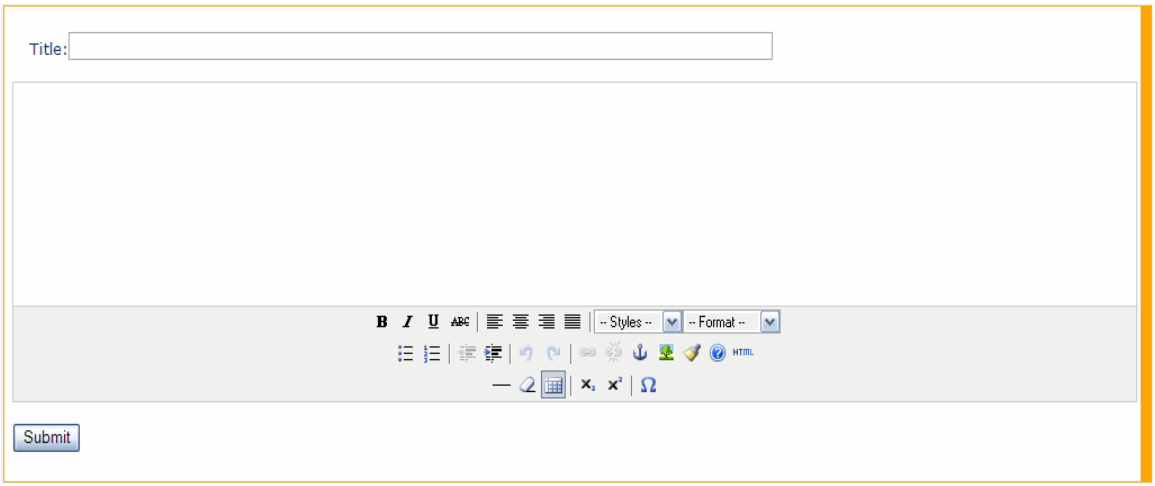

Figura 22 – Modelo "*What you see is what you get*" usado no fórum.

O modelo de dados (ver Figura 23) consiste em três tabelas relacionais em que é estabelecida a relação entre as entradas por doença e as entradas por utilizador e seus dados associados. Com este modelo simples tornou-se possível que um utilizador com um único registo possa efectuar entradas em diversos fóruns.

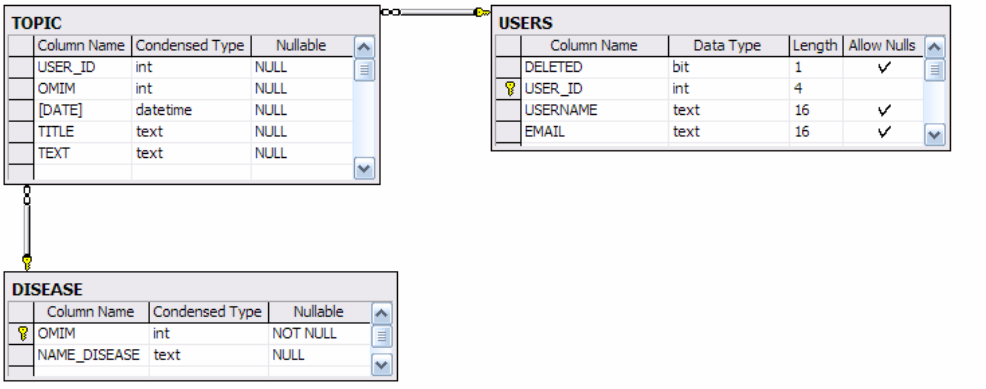

Figura 23 – Modelo de dados do fórum *DiseaseCard* .

4.2.4 O Sistema de Administração

Outro dos requisitos iniciais deste trabalho era a criação de um sistema que permitisse a administração do fórum e do portal.

Para tal foi criada uma entrada no portal para utilizadores registados com dois tipos de perfis: Administrador *Full Access*, e Administrador *Limited Access*. Ambos os perfis de utilizador foram descritos anteriormente. O Administrador *Full Access* tem a possibilidade de administrar a base de dados, através dos componentes de actualização de *MorbidMap* e da listagem de sinónimos. Tem ainda a possibilidade de gerir os *posts* do fórum, os utilizadores e também criar contas de acesso para novos administradores.

Todas as funcionalidades encontram-se disponíveis na página de administração sendo o menu do tipo acordeão (ver Figura 24), escondendo as opções conforme as funcionalidades que se desejam utilizar.

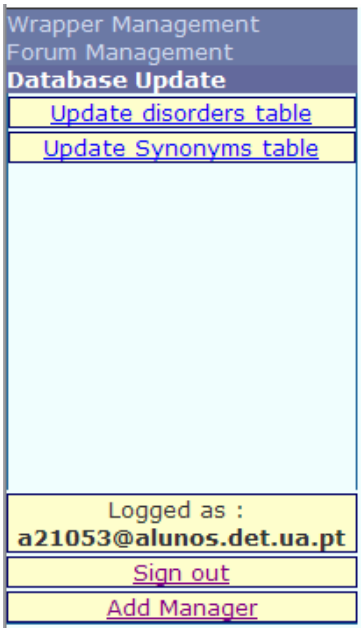

Figura 24 – Modelo de menu acordeão utilizado no sistema de administração

O Administrador *Limited Access* tem apenas permissões para aceder ao menu *Wrapper Management* e Fórum *Management*.

No menu *Wrapper Management* é possível obter uma representação em árvore do *Wrapper* de configuração do *webcrawler*. A representação do protocolo é efectuada em 3 níveis principais (ver Figura 25):

- o nome que identifica a base de dados a explorar representada por 3 cilindros;

- o endereço do recurso representado por uma página *web*;
- e o terceiro que se decompõe em dois grupos: padrão a encontrar (*Regex)* e o identificador da variável onde será armazenado (*put into)*.

Figura 25 – Representação em estrutura de árvore do XML de configuração do *Wrapper*.

Na mesma entrada é possível aceder à representação gráfica da profundidade de penetração do *wrapper*, sendo representada a percentagem de doenças que têm informação sobre cada entrada. Por exemplo, observando a Figura 26 é possível perceber que 82.18% das doenças possuem dados sobre o componente Gene.

Figura 26 – Gráfico de representação de profundidade de penetração do wrapper.

Na entrada do menu Fórum *Management* todos os administradores têm a possibilidade de visualizar as ultimas entradas nos fóruns ordenados cronologicamente. Podem

também verificar a listagem de todos os utilizadores do fórum e aceder a todos as entradas de cada um dos utilizadores conforme a Figura 27.

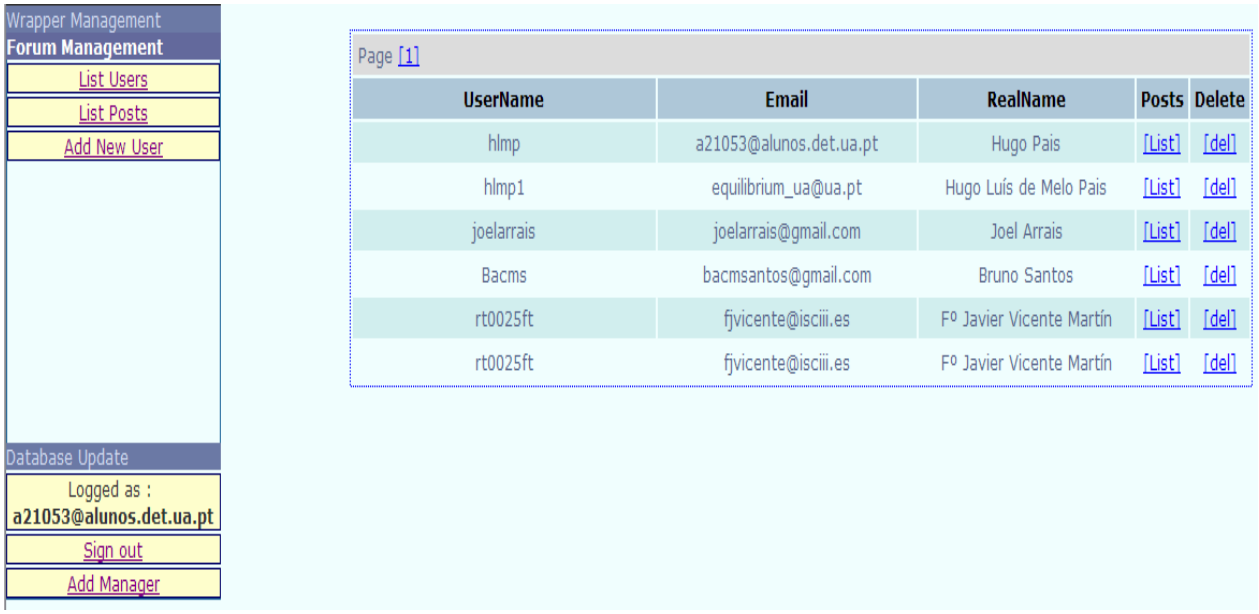

Figura 27 – Ilustração da área de gestão de utilizadores.

4.3 Sumário

Neste capítulo foi discutida a arquitectura inicial do sistema *DiseaseCard* e seu funcionamento. Foram apresentadas os objectivos e componentes desenvolvidos, as alterações ao diagrama de casos de uso assim como aos modelos de dados. Também foram abordadas as técnicas utilizadas no desenvolvimento dos sistemas de Administração, Fórum e Modelos de dados leves.

5 Conclusões

O sistema *DiseaseCard* é um portal público com capacidade de integração automática de informação do genótipo ao fenótipo, nomeadamente para doenças raras. As potencialidades do portal foram expandidas após a implementação de modelos de dados leves que melhoram consideravelmente a usabilidade do mesmo. Actualmente é possível que os utilizadores comuniquem entre si, facilitando a propagação do conhecimento científico e a comunicação entre pacientes e especialistas. Os fóruns por doença podem ser considerados um ponto de referência para pacientes e familiares, facilitando a possibilidade de interacção entre esta comunidade. Com a introdução do sistema de administração o portal apresenta um modelo transparente para análise do comportamento do *webcrawler* permitindo que o administrador efectue actualizações ao sistema sem ter que possuir conhecimentos profundos sobre o seu funcionamento. É igualmente possível moderar os fóruns, visualizar estatísticas e analisar a configuração do *wrapper* do *webcrawler*. O modelo de implementação actual, com a colaboração de especialistas com conhecimentos para definir o modelo de extracção, pode facilmente ser adaptado a outras áreas do conhecimento sem que seja necessário grande esforço de desenvolvimento, pois o modelo de dados de extracção encontra-se estável e com resultados de grande qualidade, apesar da sua aparente simplicidade.

No futuro, seria interessante que o portal fosse introduzido num ambiente clínico. O portal tem informação capaz de satisfazer diferentes tipos de utilizadores como os médicos de cuidados primários, os geneticistas ou os especialistas em doenças raras. Informação associada a sintomas, a centros de diagnóstico ou a possíveis centros e tratamentos permitem que o clínico encaminhe o paciente para um serviço especializado com a máxima celeridade. No caso do especialista ou investigador tem neste portal a oportunidade para efectuar um estudo biomédico integrado sobre cada patologia. O portal é capaz de responder a perguntas como:

- Quais as características principais da doença?
- Existem medicamentos para esta doença?

- Existem terapias genéticas disponíveis?
- Quais são os laboratórios que executam testes genéticos?
- Existem ensaios clínicos?
- Quais os genes que causam a doença?
- Em que cromossomas se localizam os genes da doença?

A capacidade de resposta a questões tão prementes como as anteriores faz com que o *DiseaseCard* se possa tornar uma ferramenta fundamental num ambiente clínico moderno, assim como em qualquer laboratório de investigação, pois permite um acesso simples, eficaz e transparente, permitindo encontrar toda a informação relevante sem que se domine as nomenclaturas de pesquisa e de navegação entre as diferentes bases de dados.

Para realizar o trabalho apresentado, foi necessário efectuar um estudo aprofundado do sistema implementado. Foi também necessário estudar a *framework Struts*, a tecnologia *Ajax* e diversos *packages* que possuíam tais funcionalidades.

De forma a realizar a manutenção ao sistema, foi necessário adquirir um grande conhecimento da estrutura das bases de dados heterogéneas usadas na integração de informação e sobre a forma como estas estão relacionadas.

6 Referências

1. O'Reilly, T., *What Is Web 2.0: Design Patterns and Business Models for the Next Generation of Software*. 2005.
2. Brown, J.S. and P. Duguid, *The social life of information*. 2000, Boston: Harvard Business School. X, 320 sider.
3. Surowiecki, J., *The Wisdom of Crowds*. 2004 Anchor; Reprint edition (August 16, 2005). 336 pages
4. Wal, T., *Folksonomy Definition and Wikipedia :: Off the Top :: vanderwal.net*.
5. Klemperer, P., *Network Effects and Switching Costs: two short essays for the new New Palgrave*, 2006-W06, Editor. 2006, Economics Group, Nuffield College, University of Oxford.
6. Nilsen//NetRatings, *Youtube U.S. Web Traffic Grows 75 Percent Week over Week*. Julho de 2006.
7. Farrell, J. and P. Klemperer, *Coordination and Lock-In: Competition with Switching Costs and Network Effects*. 2006: SSRN.
8. Ebersbach, A., M. Glaser, and R. Heigl, *Wiki : Web Collaboration*. 2005: Springer.
9. Cych, L., *Social networks*, in *Emerging Technologies for Learning*. 2006, Becta ICT Reasearch
10. Cory, D., P. Shelley, and J.S. Johnson, *Essential Blogging*. 2002: O'Reilly & Associates, Inc. 244.
11. Benkler, Y., *The Wealth of Networks : How Social Production Transforms Markets and Freedom*. 2006: Yale.
12. MILLEN, D., FEINBERG, J., KERR, B., *Social Bookmarking in the enterprise.* Pode ser encontrado em: http://www.acmqueue.com/modules.php?name=Content&pa=showpage&pid=344, 2005.

13. Lionel, F. and S. Damien, *Hands-On Guide to Video Blogging and Podcasting: Emerging Media Tools for Business Communication (Hands-on Guide)*. 2006: Focal Press.
14. Group, N.W., *The Atom Syndication Format*. 2005, The Internet Society.
15. Johnson, D. *AJAX: Dawn of a new developer. The latest tools and technologies for AJAX developers*. 2005 [cited 15 de Junho de 2007]; Available from: http://www.javaworld.com/javaworld/jw-10-2005/jw-1017-ajax.html.
16. Hinchcliffe, D. *The coming RIA wars: A roundup of the Web's new face*. Enterprise Web 2.0 2006 [cited Acedido em 15 de Junho de 2007]; Available from: http://blogs.zdnet.com/Hinchcliffe/?p=65.
17. Eichmann, D., *The {RBSE} Spider - Balancing Effective Search Against Web load.* Computer Networks and ISDN Systems, 1994.
18. Pinkerton, B. *Finding what people want: Experiences with the webcrawler*. in *International Conference on the World Wide Web*. 1994.
19. MCBRYAN, O.A. *GENVL and WWWW: Tools for taming the web*. in *First International World Wide Web Conference*. 1994. Geneva, Switzerland.
20. Burner, M. (1997) *Crawling towards eternity: Building an archive of the World Wide Web*. New Architect: Internet Strategies for Technology Leaders **Volume**,
21. Miller, R.C.B., K. SPHINX, *A Framework for Creating Personal, Site-Specific WebCrawlers.* Proceedings of the Seventh International World Wide Web Conference, 1998: p. Brisbane, Australia. 30: ISDN Systems, April 1998. p. 119-130.
22. Brin, S. and L. Page, *The anatomy of a large-scale hypertextual Web search engine.* Computer Networks and ISDN Systems, 1998. **30**(1--7): p. 107-117.
23. Allan, H. and N. Marc, *Mercator: A scalable, extensible Web crawler.* World Wide Web, 1999. **2**(4): p. 219-229.
24. Boldi, P., et al., *Ubicrawler: A scalable fully distributed web crawler*. 2002.
25. Soumen, C., *Mining the Web: Discovering Knowledge from HyperText Data*. 2002: Science \& Technology Books. 350.
26. Alexandros, N., Z. Petros, and C. Junghoo, *Downloading textual hidden web content through keyword queries*, in *Proceedings of the 5th ACM/IEEE-CS joint conference on Digital libraries*. 2005, ACM Press: Denver, CO, USA.

27. Broder, A., *Some applications of Rabin's Fingerprinting Method.* Sequences II: Methods in comunications, Security and Computer Science, Springer-Verlag, 1993: p. 143-152.
28. Robert, C.M. and B. Krishna, *SPHINX: a framework for creating personal, site-specific Web crawlers*, in *Proceedings of the seventh international conference on World Wide Web 7*. 1998, Elsevier Science Publishers B. V.: Brisbane, Australia.
29. Comission, E. *Useful Information on Rare Diseases From an EU Perspective*. Health & Consumer Protection Directorate 2004 [cited 15 de Junho de 2007]; Available from: http://ec.europa.eu/health/ph_information/documents/ev20040705_rd05_en.pdf.
30. Eurordis. *Background Paper on Orphan Diseases for the "WHO Report on Priority Medicines for Europe and the World"* 2004 [cited; Available from: www.eurordis.org.
31. Dias, G.M.S., *Sistema de Recuperação Automática de Informação Biomédica*. 2006, Universidade de Aveiro: Aveiro.